UNIVERSIDAD DE LA RIOJA

TRABAJO FIN DE ESTUDIOS

Título

Filtrado Digital de Señal con DSP

Autor/es

ÍÑIGO RODRÍGUEZ MARTÍNEZ

Director/es

ANTONIO MOISÉS ZORZANO MARTÍNEZ

Facultad

Escuela Técnica Superior de Ingeniería Industrial

Titulación

Grado en Ingeniería Electrónica Industrial y Automática

Departamento

INGENIERÍA ELÉCTRICA

Curso académico

2016-17

publicaciones.unirioja.es
E-mail: publicaciones@unirioja.es

UNIVERSIDAD DE LA RIOJA

TRABAJO FIN DE GRADO

Título
Filtrado Digital De Señal Con DSP
Autor/es
Iñigo Rodríguez Martínez
Director/es
Antonio Moisés Zorzano Martínez
Facultad
Escuela Técnica Superior de Ingeniería Industrial
Titulación
Grado en Ingeniería Electrónica Industrial y Automática
Departamento
Ingeniería Eléctrica
Curso Académico
2016-2017

Filtrado digital de señal con DSP, trabajo fin de grado de Iñigo Rodríguez Martínez, dirigido por Antonio Moisés Zorzano Martínez

UNIVERSIDAD DE LA RIOJA

ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA INDUSTRIAL

TRABAJO DE FIN DE GRADO

TÍTULO DEL TRABAJO DE FIN DE GRADO (TFG):

FILTRADO DIGITAL DE SEÑAL CON DSP

DIRECTOR DEL TFG:

ANTONIO MOISÉS ZORZANO MARTÍNEZ

AUTOR DEL TFG:

IÑIGO RODRÍGUEZ MARTÍNEZ

TITULACIÓN: **GRADO EN INGENIERÍA ELECTRÓNICA INDUSTRIAL Y AUTOMÁTICA**

DEPARTAMENTO: **INGENIERIA ELECTRICA**

CURSO ACADÉMICO: **2016/2017**

CONVOCATORIA: **2 DE MARZO**

Resumen

Este trabajo fin de grado pretende desarrollar una serie de unidades didácticas orientadas a mejorar el aprendizaje del filtrado digital de señales, aplicándolo a señales reales que el usuario puede comprobar de forma visual y audible.

Con tal fin, se ha diseñado y desarrollado una aplicación, o herramienta JAVA, FDSP (Filtrado basado en DSP), para presentar una interfaz dedicada al desarrollo de unas prácticas/experimentos de filtrado digital de señal que serán de gran ayuda para el aprendizaje de la materia. Esta herramienta otorga al alumno una comunicación intuitiva, que con su simple visualización, facilitara los pasos a seguir para realizar diferentes tareas de filtrado.

Por otro lado, se ha realizado una selección apropiada de los contenidos de filtrado digital de señales en relación con los resultados de aprendizaje esperados. Seguidamente se han diseñado y validado unos experimentos que nos permiten realizar tanto un aprendizaje más superficial (orientado a un usuario de la herramienta más ajeno al conocimiento del filtrado digital), y un aprendizaje más profundo (orientado a un usuario experimentado en temas de filtrado digital), a elección del usuario, pudiendo simplemente seguir las guías sin darle más vueltas sin realizar ningún cambio, o profundizar en el conocimiento de la herramienta y realizar otros filtrados realizando modificaciones en los experimentos dados.

Se facilita dentro de la herramienta el contenido teórico a repasar necesario para poder realizar los diversos experimentos con el conocimiento necesario para su entendimiento.

Se incluyen además guías que nos llevaran paso a paso por las diferentes secciones de la herramienta para no dar lugar a equívocos facilitando en gran medida la labor a realizar por parte del usuario.

El objetivo a alcanzar en la realización de estos experimentos, es alcanzar un equilibrio teórico/práctico que permita sacar el máximo rendimiento de las prácticas de laboratorio en la asignatura “Procesado digital”, trabajando con distintas herramientas (SoundCard Scope, Matlab, Code Composer Studio) junto con un kit de desarrollo basado en un DSP haciendo así nuestros experimentos reales.

Abstract

On one hand this Final Degree Project tries to develop a series of educational utilities focused on improving the process of learning Digital Signal Filtering.

With this goal in mind, we have designed and developed a JAVA app, to use an interface focused on a series of experiments of digital signal filtering, which will be useful to learn the contents of this subject. This tool gives the pupil an intuitive interface to do different filtering tasks.

On the other hand, we have selected contents of this subject, in relation with the results expected from the pupil learning. Some experiments have been designed and developed, which allow to have a preliminary learning of the subject (focused on using the tool with low knowledge of Digital Signal Filtering) and a deep learning (focused on an experienced user, familiar with Digital Signal Filtering), selected by the user, given the chance of just following the tutorial, or explore the subject more deeply modifying all the given experiments.

Theorical content is given within the application, to understand the experiments and all the concepts behind the theory, to ensure a full understanding of the matter.

Guides are also included, which will take us step by step through the different sections of the tool, to avoid mistakes made by the user.

The objective to reach while doing this experiments is to achieve a theoretical and practical balance, to make the most out of the practical classes of the subject, working with different tools (SoundCard Scope, Matlab, Code Composer Studio), alongside a development kit based on a DSP, to make all the experiments real.

Agradecimientos

A mis padres y a mi familia en general, gracias por vuestro apoyo incondicional en todo momento, siempre habéis estado conmigo y siempre me habéis empujado hacia adelante.

Mis sinceros agradecimientos a Antonio Moisés Zorzano Martínez que me ha guiado durante este proyecto, sin flaquear nunca en su actitud ante los diversos problemas encontrados.

Finalmente gracias a esas personas que han estado a mi lado durante la realización del grado, ya que todos ellos han aportado su granito de arena en mi formación.

UNIVERSIDAD DE LA RIOJA

ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA INDUSTRIAL

DOCUMENTO Nº 1:

Índice General

TÍTULO DEL TRABAJO FIN DE GRADO (TFG):

FILTRADO DIGITAL DE SEÑAL CON DSP

DIRECTOR DEL TFG:

ANTONIO MOISÉS ZORZANO MARTÍNEZ

AUTOR DEL TFG:

IÑIGO RODRÍGUEZ MARTÍNEZ

TITULACIÓN: **GRADO EN INGENIERÍA ELECTRÓNICA INDUSTRIAL Y AUTOMÁTICA**

DEPARTAMENTO: **INGENIERIA ELECTRICA**

CURSO ACADÉMICO: **2016/2017**

CONVOCATORIA: **2 DE MARZO**

ÍNDICES

Índice General:

ÍNDICES

ÍNDICES

Índice de Ilustraciones:

ÍNDICES

1 - INTRODUCCIÓN:

1.1 – Objetivo:

Este proyecto nace de la necesidad de complementar y asentar los conocimientos adquiridos de los alumnos de la asignatura de Procesado Digital.

El principal objetivo es otorgar al alumno una herramienta de trabajo, apoyándonos en el uso de un DSP para la realización de experimentos reales, pudiendo así poner en práctica los conocimientos adquiridos y obtener soluciones que nos permitan verificar esos conocimientos, visualizando de forma directa sus resultados.

1.2 – Antecedentes:

Las prácticas que se realizaban hasta el momento en la asignatura de Procesado digital, se realizaban mediante una herramienta software (EDSP99), cuyo objetivo era la formación en procesado digital de señal desde sus aspectos más básicos. FDSP viene a complementar la formación que se daba en EDSP99, centrado sus objetivos en el filtrado y materializándolo en su realización con un procesador digital de señal (DSP).

En segundo lugar diferenciar una simulación de un experimento real. Nadie pone en duda el uso de simulaciones durante el aprendizaje y como apoyo para el diseño de filtros que es lo que aquí nos concierne. Pero hasta que no es llevado a cabo en un experimento real, no aparecerán los verdaderos problemas a solucionar.

Por tanto si aprendemos a trabajar con estos problemas desde el minuto uno, la formación será más completa, y el alumno llevara consigo una formación más experimentada y endurecida.

1.3 – Organización y alcance del proyecto

Para la realización de este trabajo, buscando llevarlo a cabo de una forma lógica, se ha dividido en cinco fases, que se han desarrollado de forma escalonada para no perder la visión global de la herramienta en conjunto.

La primera fase ha consistido en seleccionar los contenidos teóricos de la asignatura que se van a tratar. Dichos contenidos se ha dividido en cuatro unidades didácticas de la siguiente manera:

INTRODUCCIÓN

1. **Introducción a la herramienta software de desarrollo:** el objetivo es una primera toma de contacto y una familiarización con los elementos tanto software como hardware que se van a emplear en los distintos experimentos. Además de que se plantean una serie de ejercicios a realizar por el alumno donde se ponen a prueba los conocimientos adquiridos en esta primera toma de contacto.
2. **Filtrado FIR (Respuesta impulsional finita):** con este experimento, el alumno tendrá la oportunidad de implementar en el DSP diferentes filtros FIR, que generara él mismo con ayuda de la herramienta Matlab®. Se plantean, de la misma manera, una serie de ejercicios que profundizarán en la generación de filtros FIR.
3. **Filtrado IIR (Respuesta impulsional infinita):** de la misma manera que en el experimento anterior se realizara un filtrado, pero en este caso será del tipo IIR. Se generará de la misma manera el filtro en Matlab. pero en este caso mediante la herramienta FDATool y de la misma manera tendrá sus correspondientes ejercicios.
4. **Comparación de filtros:** permite obtener resultados de filtrado de diferentes filtros generados por el alumno, tanto FIR como IIR, con lo que el alumno podrá sacar conclusiones de los resultados de filtrado y el mejor o peor comportamiento de cada uno de los filtros. Cerrando así el recorrido de filtrado.

Una vez seleccionados los contenidos a introducir, **la segunda fase** ha sido definir un entorno de trabajo basado en DSP y Matlab que nos permita estudiar todo el temario seleccionado de una manera lógica e intuitiva. Por eso y por su coste reducido, se ha elegido el kit de desarrollo de Texas Instruments TMS320VC5505 USB Stick eZdsp, que incluye licencia del IDE Code Composer Studio y nos ofrece ampliamente todo lo que necesita para realizar a cabo la labor de filtrado planteada anteriormente.

Cabe hacer hincapié en el factor económico. Por eso el sistema propuesto para el desarrollo de prácticas, no supondrá un gasto importante ni a los alumnos, ni a la universidad.

Por tanto se ofrece al alumno la posibilidad de trabajar con un entorno de procesado digital de señales, de forma que puedan ser capaces de valorar y experimentar todos los factores necesarios a la hora de poner en funcionamiento con experimentos reales, un sistema de esta naturaleza.

A continuación para el correcto desarrollo de las prácticas de laboratorio serán necesarios algunos elementos tanto hardware como software. Algunos de ellos han

sido nombrados, pero a continuación se verá cada uno de manera breve, ya que más adelante encontrarán una descripción más detallada de todos ellos:

- El elemento principal sobre el que se ha basado el trabajo realizado en los experimentos, es un kit de desarrollo fabricado por Texas Instruments: **TMS320VC5505 USB Stick eZdsp**, la tarjeta incluida en el kit será alimentada directamente a través de un puerto USB del PC, o un cable USB macho-hembra que nos permitirá que sea realmente fácil el trabajo con ella, ya que tanto la alimentación, la carga de los programas, y la entrada/salida de las señales a filtrar se realizara en este mismo puerto USB.
  Además la tarjeta contendrá dos puertos de audio estéreo que se podrán utilizar en el caso de ampliar los experimentos e introducir audio por uno de los puertos de forma constante y escuchar la salida a tiempo real.
- Junto con el kit de desarrollo, se incluirá el **IDE Code Composer Studio**, software perteneciente a Texas Instruments. Aprovechando esta licencia, utilizaremos esta herramienta para la programación y configuración del DSP y sus periféricos.
- Otra de las herramientas utilizadas, es **SoundCard Scope**, se trata de un software libre para uso educativo que implementa mediante la tarjeta de sonido de nuestro PC un generador de señales de audio (reales), un analizador de frecuencias, un osciloscopio además de algunas otras funciones.
- Con todo esto solo falta de presentar una herramienta que nos ayude a la hora de calcular los filtros con sus coeficientes, tanto FIR como IIR, además de realizar otras funciones de generación de señal en uno de los experimentos y una herramienta de apoyo en el análisis de los resultados obtenidos.
  Esta herramienta no podría ser otra que **Matlab**, que cuenta con una Toolbox específica para la generación de filtros. Además nos otorgará compatibilidad con el software de Texas Instruments a la hora de exportar los coeficientes.

La tercera fase ha consistido en la programación del DSP, la generación de los experimentos que van a funcionar en nuestro kit de desarrollo. Esta fase se dividió en dos partes: la primera era comprobar que todo lo planteado inicialmente era posible realizarlo, como podría ser sustituir los coeficientes del tipo de filtrado a llevar a cabo, sin tener que modificar nada más en el programa. La segunda parte era la entrega de unos experimentos finalistas, de tal manera que en las distintas prácticas, el alumno podría centrarse en el diseño de filtros y el análisis de los resultados, sin malgastar su tiempo resolviendo cuestiones relativas a la programación del DSP.

Siempre quedara abierto este camino para aquel que quiera desarrollar sus conocimientos de manera independiente.

INTRODUCCIÓN

La cuarta fase se centra en la practicas a realizar en si mimas. La generación de documentos teóricos, scripts, sesiones de Matlab, con ejemplos de uso además de filtros generados por defecto que el alumno pueda usar en un primer momento.

Se incluirán también guías de cada uno de los experimentos para facilitar el seguimiento de los experimentos, y una puesta en marcha inicial para el alumno y que este sea capaz de recoger la metodología de trabajo y seguir a continuación de forma independiente explotando los recursos que nos ofrece la herramienta.

En la quinta fase y por último lugar, para intentar recoger todas las herramientas utilizadas en un entorno que simplifique el seguimiento de los experimentos, ya que entiendo que los diferentes cambios entre los programas a utilizar para un usuario ajeno al proceso de filtrado, podría dar lugar a equívocos.

Por esta razón, se ha generado una interfaz en **Java** mediante **NetBeans**, que recogerá todos los puntos comentados anteriormente y dará un soporte sólido para la ejecución de los experimentos. Se podrá realizar un seguimiento visual del proceso a seguir en nuestra labor, a través del entorno generado.

2 – DESCRIPCIÓN DEL HARDWARE Y SOFTWARE EMPLEADO:

En este punto se realizará de forma detallada una descripción de todos los elementos que se han puesto en marchar para alcanzar todos y cada uno de los objetivos marcados inicialmente.

En primer lugar se ha estudiado el hardware empleado, prestando especial atención en aquellos elementos que se consideran esenciales y que tengan más peso en nuestros objetivos. En segundo lugar se han revisado todos los elementos de software utilizados de la misma manera, prestando más atención a los partes más interesantes relacionadas con nuestra labor.

2.1 - Análisis del hardware empleado.

El hardware seleccionado para la realización del proyecto ha sido el kit de desarrollo de Texas Instruments, TMS320VC5505 USB Stick eZdsp. Se trata de un kit de bajo coste que lo convierte en el kit ideal para nuestro propósito. La alimentación de la tarjeta se realiza vía USB, por lo que no se necesitará ninguna fuente de alimentación externa, bastará con conectarla al puerto USB del ordenador. El kit incluye la licencia del IDE Code Composer Studio, del que se hablará más adelante. Servirá para programar y configurar nuestro DSP. Esto abrirá todo el abanico de posibilidades para trabajar y experimentar con uno de los DSP de 16 bits de bajo consumo presentes en la industria actualmente: C5505.

Ilustración 1: ezDSP TMS320VC5505

Descripción del hardware y software empleado

Esta herramienta de bajo coste pondrá al alcance de los alumnos la realización de una fácil y rápida evaluación de las avanzadas posibilidades que un procesador como el C5505 nos puede otorgar.

La tarjeta en la que se encuentra incluido, confiere al DSP C5505 varios módulos periféricos como son el conector de expansión de pines, un puerto USB 2.0, un conjunto de LEDs y un códec de audio programable de 32 bits modelo TLV320AIC3204 de Texas Instrumentes. Este último ofrece dos puertos entrada/salida dos conectores estéreo de audio tipo mini Jack o TLR, a través de los cuales se podrá introducir las señales generadas y recibir las señales procesadas.

Además la tarjeta dispone de un bus serie que permite la conexión con un equipo host para transmitir a través del emulador embebidos del XDS100 el programa de carga y depuración del DSP.

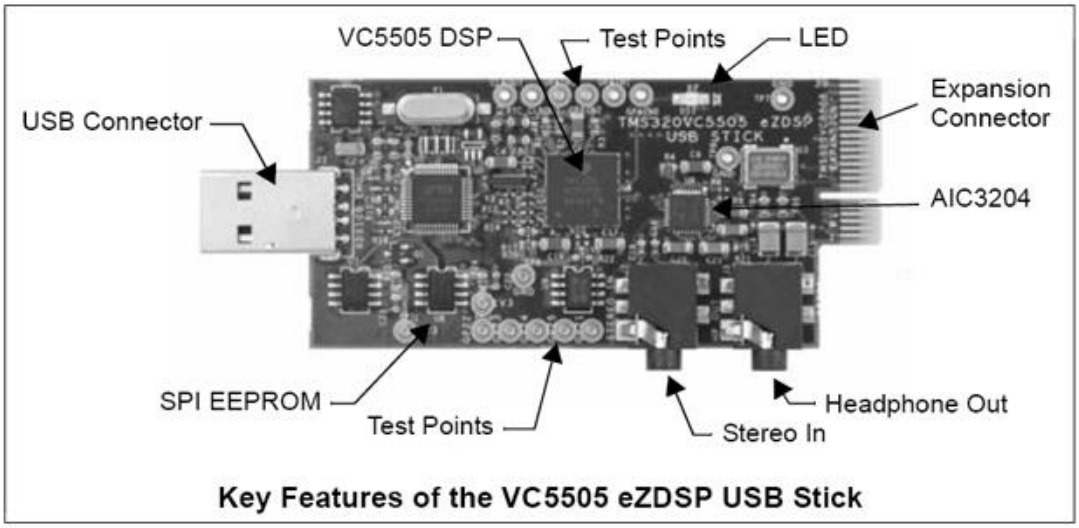


Ilustración 2: Detalle de la tarjeta de desarrollo TMS320C5505

A continuación se realizara una breve descripción de los dos elementos principales de esta tarjeta, el procesador DSP C5505 y el CODEC de audio TLV320AIC3204

2.1.1 - C5505

El C5505 es uno de los miembros de la familia de procesadores TMS320C5000 de coma fija diseñados por Texas Instruments para realizar aplicaciones de bajo consumo. Los DSp de coma fija están basados en la generación de procesadores TMS320C55XX.

La arquitectura de la familia de DSPs C55XX logra un alto rendimiento manteniendo un bajo consumo. Esto se consigue al aumentar el paralelismo computacional y nunca perdiendo de vista la finalidad de mantener el consumo bajo.

Descripción del hardware y software empleado

La CPU posee un estructura interna de buses que se compone por un bus de programa, un bus de lectura de datos de 32 bit, dos buses de lectura de datos de 16 bit, dos buses de escritura de datos de 16 bit, y también algunos buses adicionales dedicados a los periféricos y al DMA (siendo este el encargado de la lectura escritura de los datos en el CODEC). Estos buses ofrecen la posibilidad de realizar hasta cuatro lecturas y dos escrituras de datos de 16 bit en un único ciclo. El dispositivo incluye además cuatro controladores DMA, cada uno con cuatro canales proporcionando la posibilidad de mover datos por dieciséis canales independientes sin la necesidad de la actuación de la CPU.

Ilustración 3: CPU

Cada controlador DMA puede realizar una transferencia de datos de 32 bits por ciclo, en paralelo y de manera independiente de la actividad de la CPU.

La CPU C55x dispone de dos unidades multiplicadoras-acumuladoras (MAC), cada una capaz de realizar multiplicaciones de 17 bit x 17 bit y una suma de 32 bit en un único ciclo. Dispone además de una unidad aritmético lógica (ALU) de 40 bits que se apoya en otra de 16 bit. El uso de las ALUs se realiza mediante un conjunto de instrucciones de control, que proporcionan la habilidad de realizar operaciones en paralelo con un bajo consumo de energía. Estos recursos son manejados por la unidad de direcciones (AU) y la unidad de datos (DU) de la CPU C55x.

La CPU C55x soporta un conjunto de instrucciones de ancho variable para poder optimizar la densidad del código. La unidad de instrucciones (IU) realiza desplazamientos de 32 bit de programa desde la memoria, interna o externa, y la cola de instrucciones hacia la unidad de programa (PU). La unidad de programa decodifica las instrucciones, dirige las tareas de la unidad de direcciones (AU) y los recursos de la unidad de datos (DU), y gestiona el *pipeline.*

La comunicación serie es soportada a través de dos periféricos *MultiMediaCard/Secure Digital* (MMC/SD), cuatro módulos Inter-IC Sound (I2S Bus), una Serial-Port Interface (SPI) con hasta 4 chips de selección, una interfaz I2C y una interfaz (UART) Universal Asynchronous Receiver/Transmitter.

Descripción del hardware y software empleado

El conjunto de periféricos del dispositivo incluye un interfaz de memoria externa (EMIF) que proporciona acceso sin colas a memorias de tipo asíncrono como las EPROM, NOR, NAND y SRAM. Además incluye algunos periféricos adicionales como un puerto USB de alta velocidad 2.0, un reloj a tiempo real (RTC). Además de tres timers de propósito general con un timer-watchdog configurable, y un (APLL) analog phase-locked loop generador de reloj.

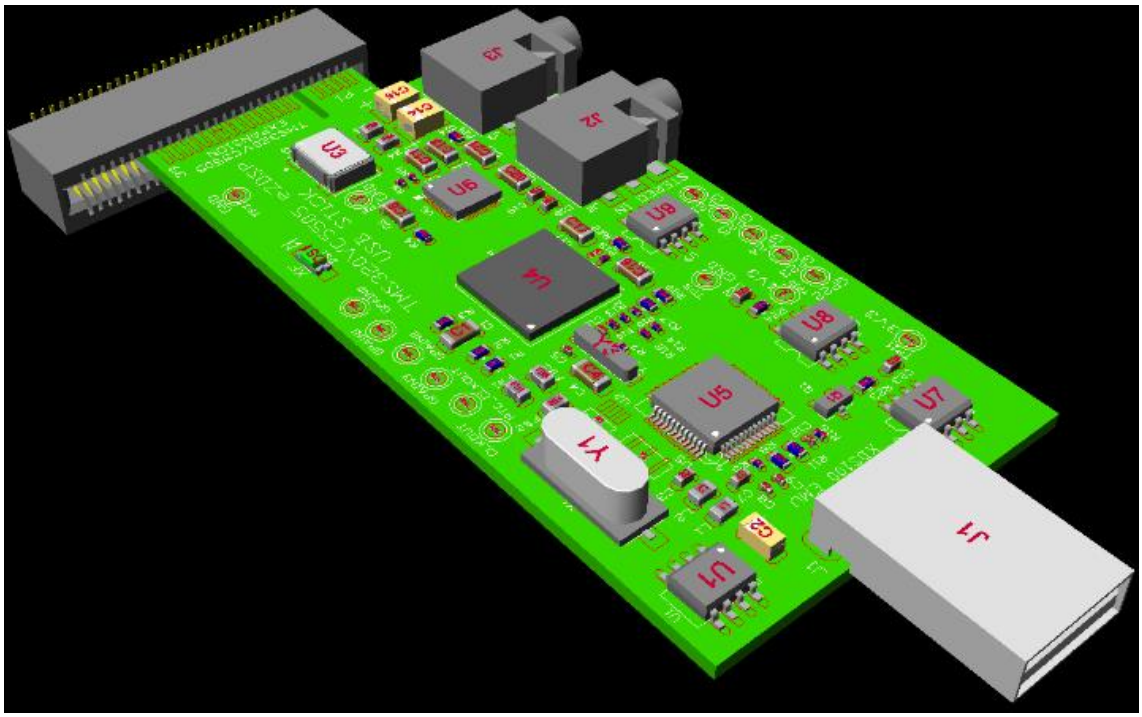


Ilustración 4: Organización Periféricos

Descripción del hardware y software empleado

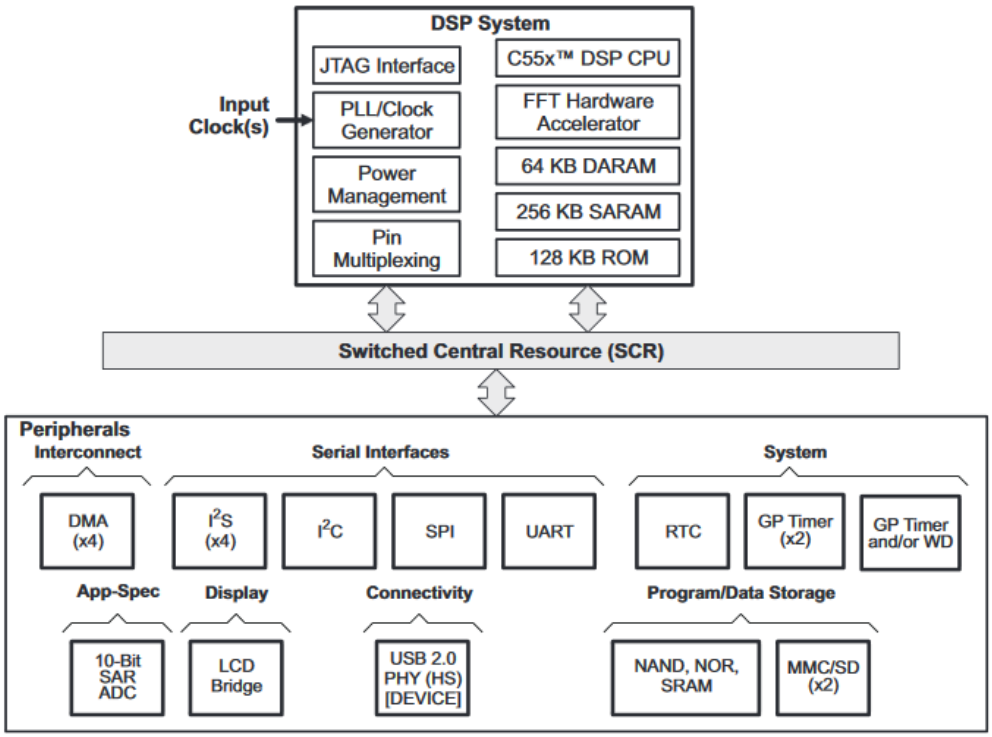


Ilustración 5: Diagrama de bloques del TMS320C5505

Adicionalmente incluye un acelerador hardware FFT que soporta valores reales y complejos desde 8 hasta 1024 puntos (en potencias de 2).

Este procesador es soportado por múltiples entornos de desarrollo como eXpressDSP o CodeComposer Studio. Incluyendo este último, herramientas para la generación de código, incluyendo un compilador de C, drivers para los dispositivos de emulación RTDX, XDS100, XDS510 y XDS560, y algunos módulos de evaluación. Puede ser utilizado con la librería C55x DSP que cuenta con más de 50 kernels software (filtros FIR, filtros IIR, FFTs y múltiples funciones matemáticas) así como librerías soportadas por el dispositivo.

Descripción del hardware y software empleado

2.1.2 – TLV320AIC3204

El TLV320AIC3204 (también conocido como el AIC3204) es un códec de audio estéreo de bajo consumo, que requiere tensiones bajas de alimentación. Tiene entradas y salidas programables, incluye tecnología PowerTune, bloques de procesamiento de señales fijos, predefinidos y parametrizables, un PLL integrado, LDOs integrados(que son reguladores lineales de voltaje DC) e interfaces digitales flexibles.

Este códec está basado en un registro extensible que se encarga del control del consumo, la configuración de los canales de entrada salida, las ganancias, los efectos, la multiplexación de pines y los relojes, dando al dispositivo la máxima precisión en la ejecución de sus aplicaciones. Combinado lo anteriormente mencionado con la avanzada tecnología PowerTune, este dispositivo puede realizar operaciones, desde reproducciones de voz mono de 8Khz, hasta reproducciones de audio estéreo de 192Khz, haciéndolo ideal para ser utilizado en reproductores de audio portátiles o para aplicaciones de telefonía.

El path (ruta) de grabación del TLV320AIC3204 puede realizar grabaciones de audio desde mono a 8 KHz hasta estéreo a 192 KHz, y posee diferentes configuraciones programables del canal de entrada que pueden ser de tipo diferencial o de tipo single-ended, así como señales de entrada flotantes o mezcladas. También incluye un preamplificador estéreo de micrófono, controlado digitalmente y un bias de micrófono integrado. Contiene además bloques de procesado de señales digitales, capaces de eliminar el ruido causado por un acoplamiento mecánico como por ejemplo el zoom óptico de una cámara digital.

El path (ruta) de reproducción ofrece bloques de procesamiento de señales con los que filtrar y generar efectos. Da la posibilidad de realizar mezcla de señales de entrada analógicas y DAC (Digital Analog Converter). Además tiene controladores de volumen programables. El path de reproducción contiene dos controladores de salida de alta potencia, así como dos salidas totalmente diferenciales. Las salidas de alta potencia se pueden configurar de varias formas, incluyendo estéreo y mono BTL.

La tecnología integrada PowerTune permite al dispositivo ser ajustado para un consumo mínimo sin perder en el camino su rendimiento. Las aplicaciones móviles tienen múltiples casos donde es fundamental un bajo consumo de energía. Por el contrario cuando el dispositivo se utiliza mediante una fuente de energía externa el consumo del dispositivo queda en un papel secundario, mientras que minimizar el ruido se vuelve más importante. Mediante el PowerTune el TLV320AIC3204 puede ser utilizado en ambas situaciones.

Descripción del hardware y software empleado

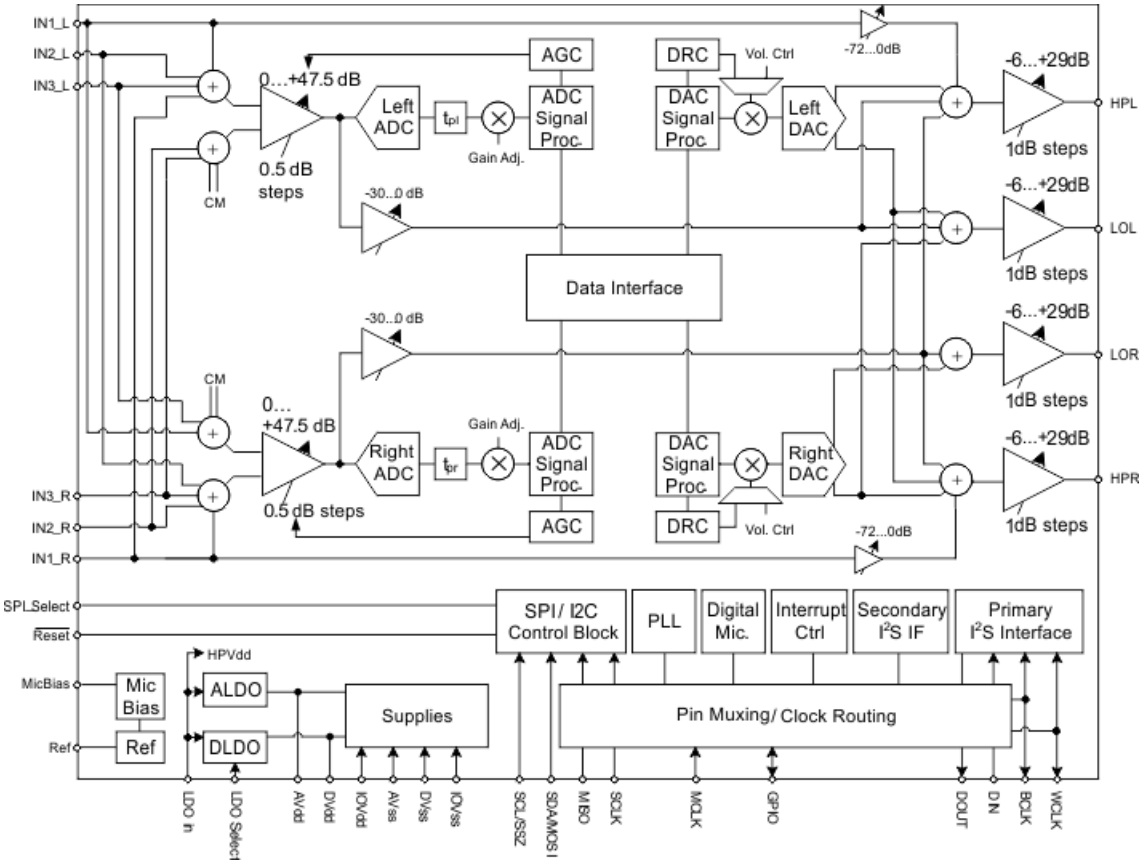


Ilustración 6: Esquema de bloques de TLV320AIC3204

Los rangos de alimentación para el TLV320AIC3204 son de 1.5 V a 1.95 V para analógica, y de 1.26 V a 1.95 V para digital. Para facilitar el diseño a nivel de sistema, el códec integra una serie de LDOs que son utilizados para generar una apropiada alimentación tanto analógica como digital partiendo de un rango de tensiones de entre 1.8 V y 3. 6V. Para las entradas-salidas digitales son soportadas tensiones de 1.1 V a 3.6V.

El reloj interno requerido por el TLV320AIC3204 puede derivarse de varias fuentes, incluyendo el pin MCLK (Master Clock), el pin BCLK (Bus de datos serie de audio con bit de reloj), el pin GPIO Entrada/salida de propósito general) o la salida del PLL interno, donde la entrada del PLL puede derivar otra vez del pin MCLK, o de los pines BCLK o GPIO. Además utilizando el PLL aseguramos la disponibilidad de una señal de reloj adecuada, no es recomendado para configuraciones de bajo consumo. El PLL es altamente programable pudiendo aceptar señales de reloj de entrada desde los 512Khz hasta los 50Mhz.

Descripción del hardware y software empleado

Con esta información damos por terminada la descripción del hardware empleado. Para llevar a cabo un estudio más en profundidad recomendamos dirigirse a la página web de Texas Instruments donde encontraremos la documentación y hojas de características (datasheets) que hacen referencia a los distintos elementos de hardware que hemos utilizado.

Dichas fuentes se incluirán en la bibliografía.

2.2 - Análisis del software empleado.

Para realizar toda la preparación del proyecto, se ha necesitado utilizar cuatro programas: Matlab, Netbeans, Code Composer Studio v.4.0 y Soundcard Scope. A continuación se analizará cada uno de ellos más detalladamente, aunque no se hará de forma exhaustiva ya que no es la finalidad de este proyecto.

2.2.1 – Matlab:

MATLAB (abreviatura de ***MAT****rix* ***LAB****oratory*, "laboratorio de matrices") es una herramienta de software matemático que ofrece un entorno de desarrollo integrado (IDE) con un lenguaje de programación propio (lenguaje M). Está disponible para las plataformas Unix, Windows, Mac OS X y GNU/Linux.

Entre sus prestaciones básicas se hallan: la manipulación de matrices, la representación de datos y funciones, la implementación de algoritmos, la creación de interfaces de usuario (GUI) y la comunicación con programas en otros lenguajes y con otros dispositivos hardware. El paquete MATLAB dispone de dos herramientas adicionales que expanden sus prestaciones, a saber, Simulink® (plataforma de simulación multidominio) y GUIDE (editor de interfaces de usuario - GUI). Además, se pueden ampliar las capacidades de MATLAB y las de Simulink con los *paquetes de bloques* (*blocksets*) y con las *cajas de herramientas* (*toolboxes*) siendo una de estas la que emplearemos conocida como FDATool

Es un software muy usado en universidades y centros de investigación y desarrollo. En los últimos años ha aumentado el número de prestaciones, como la de programar directamente procesadores digitales de señal o crear código VHDL.

La plataforma de MATLAB está optimizada para resolver problemas de ingeniería y científicos. El lenguaje de MATLAB, basado en matrices, es la forma más natural para expresar las matemáticas computacionales.

Descripción del hardware y software empleado

Los gráficos integrados facilitan la visualización de los datos y la obtención de información a partir de ellos. Una vasta librería de toolboxes preinstaladas le permiten empezar a trabajar inmediatamente con algoritmos esenciales para su dominio. El entorno de escritorio invita a experimentar, explorar y descubrir. Todas estas herramientas y prestaciones de MATLAB están probadas y diseñadas rigurosamente para trabajar juntas. Aprovecharemos esta alta compatibilidad entre las herramientas incluidas en nuestros experimentos.

En cuanto al entorno, se trata de una sencilla ventana donde se introducen comandos en modo texto y se podrán visualizar los resultados. Las representaciones gráficas se abrirán en ventanas independientes dando posibilidades a la hora de mostrar y comparar resultados.

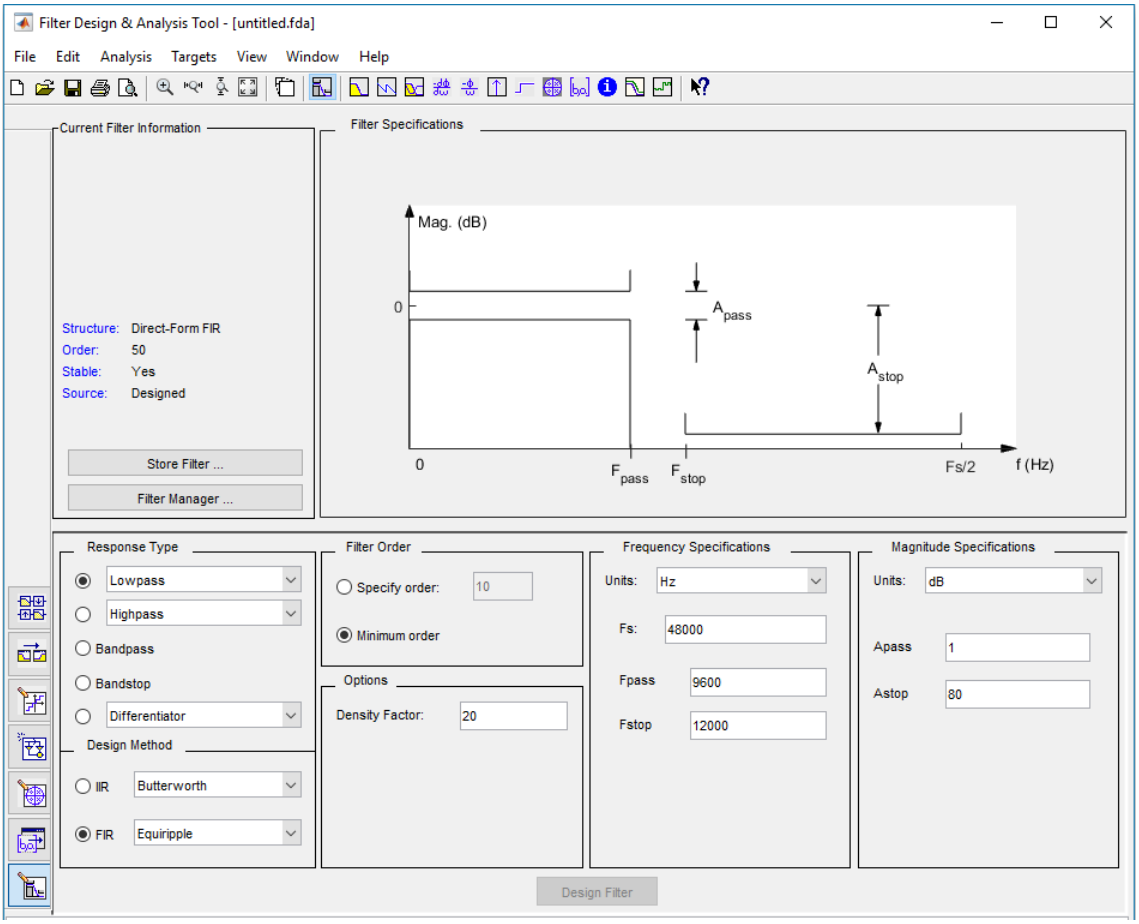


Ilustración 7: FDATool

Descripción del hardware y software empleado

La Toolbox en la que nos centraremos FDATool (FilterDesign and Analysis Tool) se utilizara para realizar el diseño de filtros durante el desarrollo de los dos últimos experimentos. Los motivos fundamentales por los que ha sido escogida es por su facilidad de uso, siendo muy intuitiva y organizada y en segundo lugar la alta compatibilidad a la hora de exportar los coeficientes de los filtros generados a un fichero .h compatible con Code Composer Studio.

2.2.2 – Netbeans:

NetBeans es un entorno de desarrollo integrado libre, hecho principalmente para el lenguaje de programación Java. Existe además un número importante de módulos para extenderlo. NetBeans IDE es un producto libre y gratuito sin restricciones de uso.

Sun MicroSystems fundó el proyecto de código abierto NetBeans en junio de 2000 y continúa siendo el patrocinador principal de los proyectos (Actualmente Sun Microsystems es administrado por Oracle Corporation).

La plataforma NetBeans permite que las aplicaciones sean desarrolladas a partir de un conjunto de componentes de software llamados *módulos*. Un módulo es un archivo Java que contiene clases de java escritas para interactuar con las APIs de NetBeans y un archivo especial (manifest file) que lo identifica como módulo. Las aplicaciones construidas a partir de módulos pueden ser extendidas agregándoles nuevos módulos. Debido a que estos pueden ser desarrollados independientemente, las aplicaciones basadas en la plataforma NetBeans pueden ser extendidas fácilmente por otros desarrolladores de software.

El NetBeans IDE soporta el desarrollo de todos los tipos de aplicación Java (J2SE, web, EJB y aplicaciones móviles). Entre sus características se encuentra un sistema de proyectos basado en Ant, control de versiones y refactoring.

A partir de NetBeans IDE 6.5.2, extiende las características existentes del Java EE (incluyendo Soporte a Persistencia, EJB 3 y JAX-WS). Adicionalmente, el NetBeans Enterprise Pack soporta el desarrollo de Aplicaciones empresariales con Java EE 5, incluyendo herramientas de desarrollo visuales de SOA, herramientas de esquemas XML, orientación a web servicies (for BPEL), y modelado UML. El NetBeans C/C++ Pack soporta proyectos de C/C++, mientras el PHP Pack, soporta PHP 5.

Destaca por su modularidad, todas las funciones del IDE son provistas por módulos. Cada módulo provee una función bien definida, tales como el soporte de Java, edición, o soporte para el sistema de control de versiones. NetBeans contiene todos los

Descripción del hardware y software empleado

módulos necesarios para el desarrollo de aplicaciones Java en una sola descarga, permitiéndole al usuario comenzar a trabajar inmediatamente.

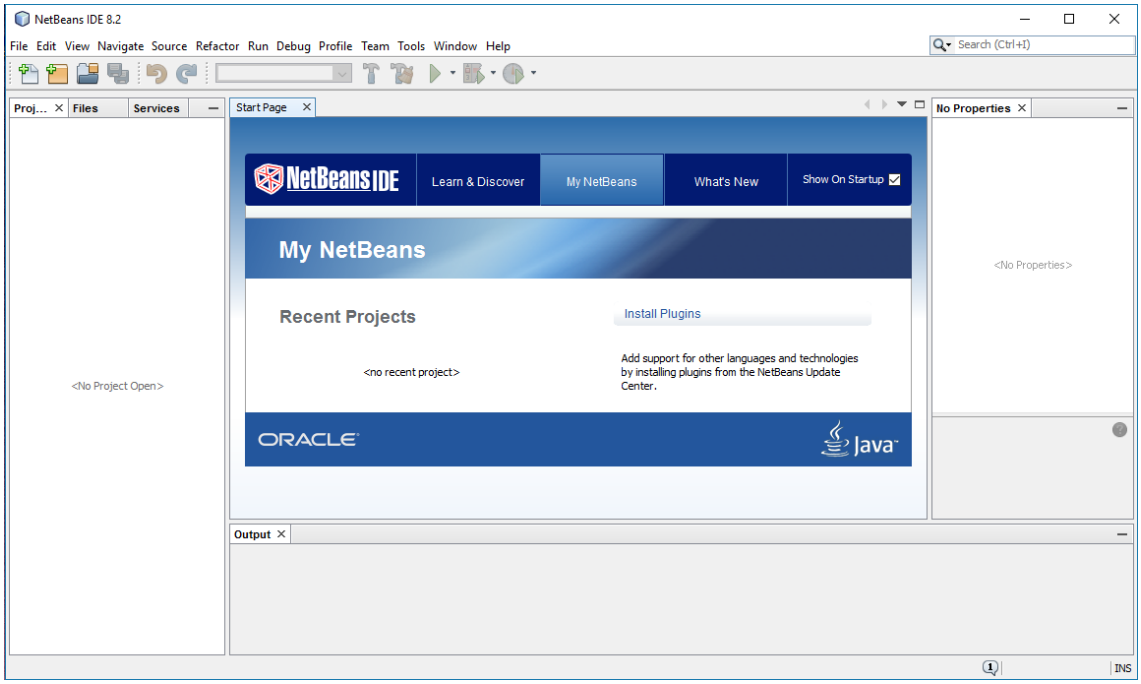


Ilustración 8: Pantalla principal Netbeans

Debido a su uso libre y su comodidad de uso, se convierte en la mejor opción para realizar la interfaz que envuelva todo el proyecto conteniendo a los distintos programas, ejerciendo de lanzador y ofreciéndonos un soporte y organizando debidamente los experimentos

2.2.3 – Code Composer Studio:

Se trata de un entorno software de desarrollo integrado (IDE) que da soporte a toda la familia de microcontroladores, sistemas embebidos y procesadores integrados de Texas Instruments.

La herramienta contiene las herramientas necesarias para la programación del DSP a utilizar utilizando el lenguaje de programación C, incluyendo todo lo necesario para su programación: Posee un compilador de C/C++ optimizado, un editor de código fuente donde desarrollar la programación, un entorno de compilación, un depurador además de muchas otras herramientas.

Se trata de una aplicación con una interfaz muy organizada e intuitiva que te guiara en los pasos de desarrollo de las aplicaciones y la depuración y carga de las mismas en el dispositivo.

Descripción del hardware y software empleado

Está inspirado en la plataforma de programación software Eclipse, pero incluyendo las mejoras más avanzadas en aplicaciones embebidas desarrolladas por Texas Instruments, dando lugar a un entorno con un fácil comienzo para empezar a escribir código, y un gran abanico de posibilidades para los desarrolladores de sistemas.

Durante los experimentos, los programas desarrollados en CCS, estarán orientados al usuario más avanzado, permitiendo al usuario iniciado un corto tránsito a través de la herramienta, simplemente ejecutando el programa y visualizando los resultados.

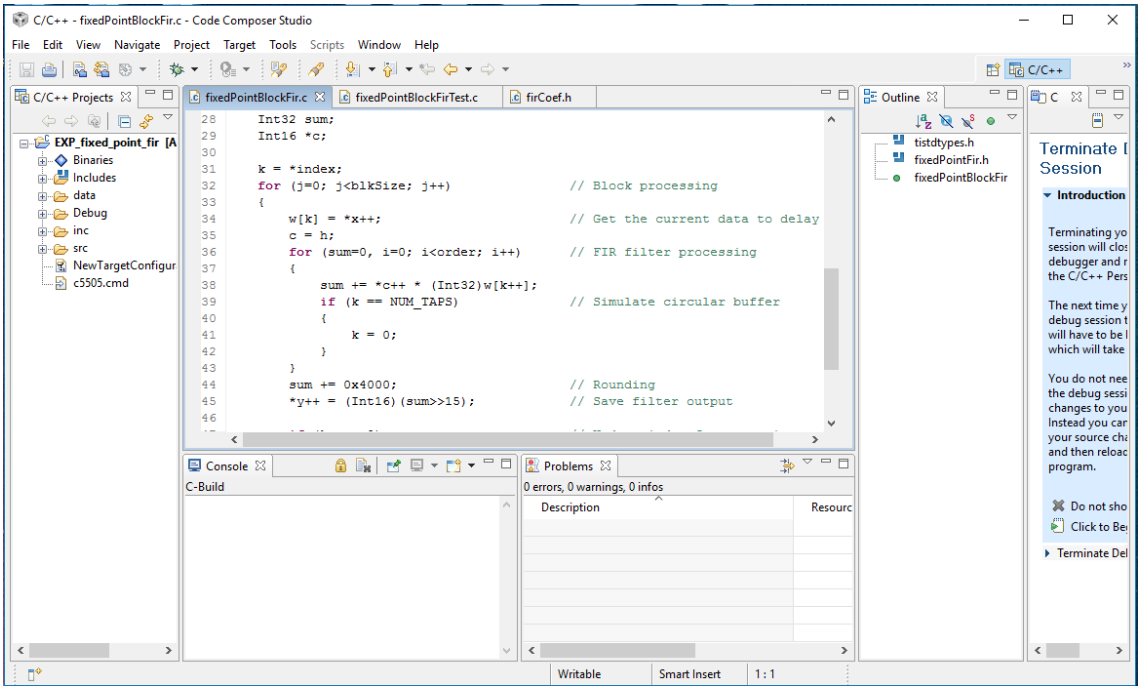


Ilustración 9: Interfaz CCS

Para el usuario avanzado contendrá gran cantidad de opciones, sencillez a la hora de leer variables o hacer representaciones gráficas para ver si las señales se están procesando de forma correcta, y la posibilidad de pausar el programa para realizar comprobaciones aunque este éste funcionando en la tarjeta.

Descripción del hardware y software empleado

2.2.4 – SoundCard Scope:

Se trata de una herramienta software de libre distribución para uso con fines educativos.

Basa su funcionamiento en la tarjeta de sonido del Pc donde se ejecuta, utilizándola como como interfaz de entradas/salidas. Recibe los datos de la tarjeta con una frecuencia de 44.1 KHz y una resolución de 16 bits. Puede ajustar el origen de datos desde las opciones de Windows.

En cuanto a su funcionamiento, permitirá más o menos prestaciones en función de la tarjeta de sonido que posea el pc, ya que se basa en ella para la generación, pero con cualquier tarjeta más o menos actual permitirá un rango de frecuencias de entre 0 Hz a 20 KHz.

El programa contiene diversas pestañas para separar las diversas utilidades.

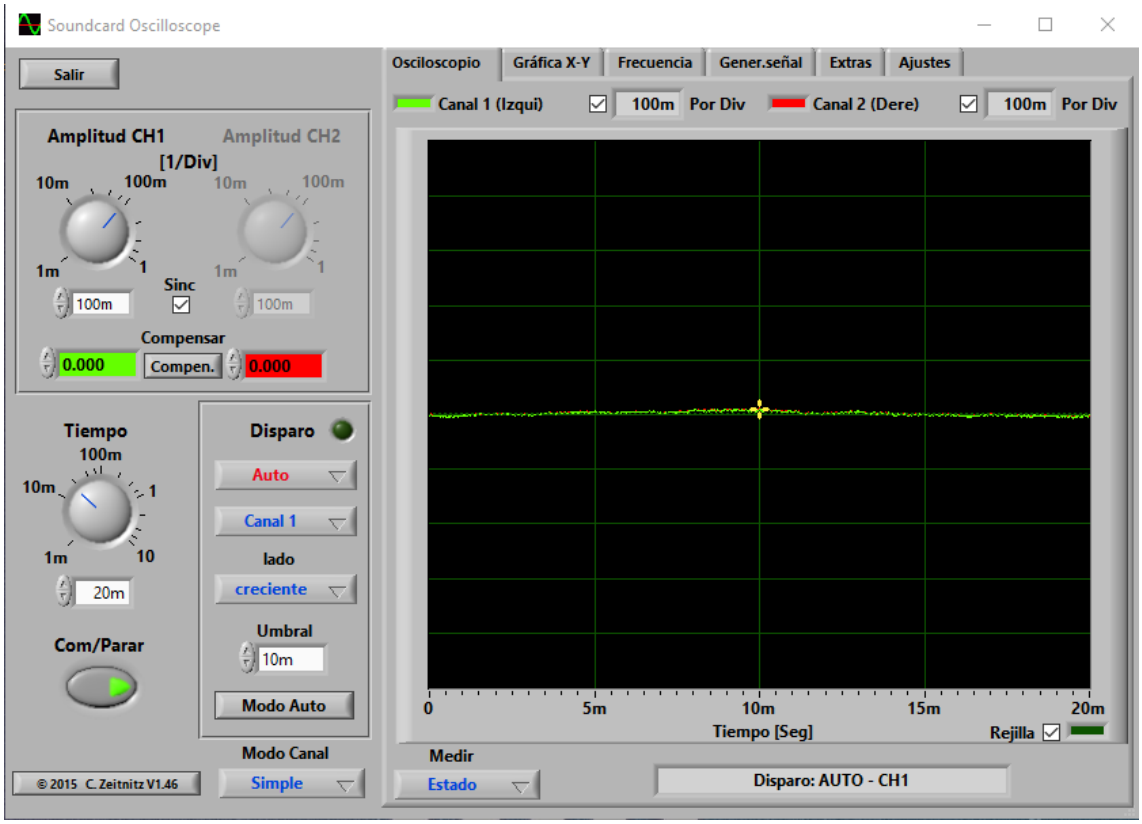


Ilustración 10: Osciloscopio SCS

Descripción del hardware y software empleado

La primera utilidad que se presenta es un osciloscopio, que incluirá todo el funcionamiento básico a encontrar en un osciloscopio digital para realizar medidas y comprobaciones que se consideren oportunas. Se podrá ajustar la amplitud, así como el tiempo, para ver las representaciones de la forma más cómoda posible y se podrán realizar mediciones colocando unos cursores, dando los valores de tensión y frecuencia. Incluirá diferentes modos de disparo que permitirán representar los dos canales que maneja de forma simultánea, individual, y realizar operaciones entre ambos.

Además del osciloscopio también incluye un analizador de espectros frecuenciales muy útil en los experimentos pudiendo ver la retención de pico o comprobar la frecuencia a la que está muestreando nuestro códec. También incluye un módulo de representación X/Y para representaciones Lissajous.

Las dos últimas funcionalidades que se utilizarán de manera conjunta como experimento inicial serán el generador de señales y el grabador de señales. Se trata de un generador de dos canales que puede generar ondas senoidales, cuadradas, dientes de sierra…, o introducirlas por medio de una formula, en un rango de frecuencia de 0 a 20 KHz y una amplitud máxima de 1 V.

El grabador de señales permite realizar diferentes tipos de grabaciones, clicando mediante botón, o utilizar un disparador tanto automático como manual.

Además proporciona ajustes en la configuración de la grabación pudiendo seleccionar la longitud de la grabación y el tiempo de pre-disparo.

3 – DISEÑO E IMPLEMENTACIÓN DE LAS UNIDADES DIDÁCTICAS:

Como se ha comentado en el punto uno de este documento, el objetivo principal es el diseño de cuatro unidades didácticas, mediante las cuales el alumno pueda acompañar su aprendizaje en la asignatura de procesado digital.

Cada uno de estos experimentos tendrá una finalidad propia en el aprendizaje de forma que los contenidos y la dificultad se van adaptando según el desarrollo de las mismas, dando conceptos muy básicos al principio y entrando en materia más consistente al final.

En este punto se hablará sobre los pasos seguidos para llevar a cabo la definición e implementación de las diferentes prácticas, analizando cada una de ellas de forma detallada.

3.1 – Práctica 1: Introducción:

Los objetivos perseguidos en la primera unidad didáctica son los enumerados a continuación:

- Familiarización con la tarjeta de desarrollo TMS320VC5505 y ejecución sobre la misma un programa básico (‘Hola Mundo’), para adquirir la metodología básica.
- Conocimiento general de funcionamiento de Code Composer Studio (CCS) incluyendo compilación, depuración y carga del programa.
- Familiarización con un osciloscopio digital, y en profundidad la generación de señales a través de la tarjeta de sonido (Sound Card Scope).
- Iniciación y uso general de Matlab, revisión de conocimientos básicos.
- Adquisición de la metodología a seguir en el uso de la herramienta final, con el modelo información, guía y pasos a seguir.

Para la consecución de estos objetivos, se hace pasar al alumno por un documento de información donde se encuentra una descripción del software a emplear, con todos los programas mencionados en los objetivos (Matlab, Code Composer Studio, Sound Card Scope), con una breve descripción individual de cada uno de ellos para el conocimiento básico de que son y para que uso están orientados.

Diseño e implementación de las unidades didácticas

A continuación, en el documento de información, se encuentra una visión general de la herramienta con las cuatro unidades didácticas, y para qué han sido orientadas, lo que ayudará al alumno a orientar su enfoque de que es lo que se va a realizar.

Y por último, una breve descripción del Hardware empleado en la práctica, TMS320VC5505 que se podrá ampliar con ayuda de las hojas de características del fabricante.

Tras esto, se encontrará un documento guía, que hará las veces de tutorial, ya que la lectura se deberá realizar durante el desarrollo de la práctica.

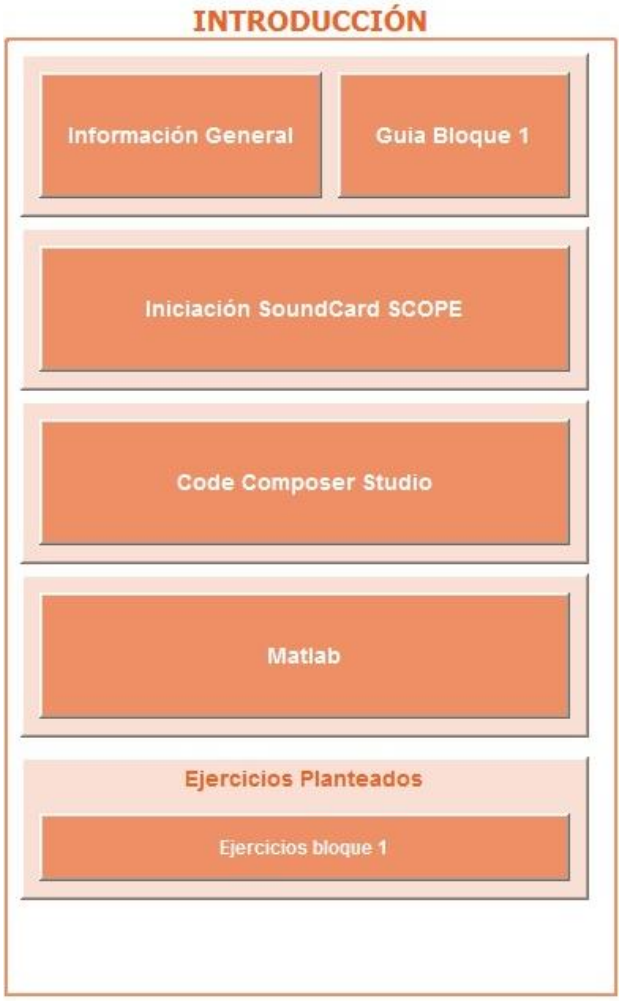


Ilustración 11: Cuadro de la herramienta, práctica 1

Este documento nos guiará paso a paso a través de las diferentes partes de la unidad didáctica uno recorriendo todos los programas utilizados, y realizando todos los ejercicios necesarios para la consecución de los objetivos.

Diseño e implementación de las unidades didácticas

En primer lugar se aprenderá a generar señales en SoundCard Scope y como grabarlas por medio de la misma herramienta. En segundo lugar, se habituará al alumno en la creación de un proyecto en Code Composer Studio, entendiendo los diferentes tipos de archivo y la carga de este programa en la tarjeta, integrándola así en nuestro entorno de trabajo por primera vez. En tercer lugar se realizará un sencillo ejercicio totalmente guiado en Matlab donde se comprenderá su funcionamiento, conociendo donde se ubican las variables, donde se visualizan los resultados…

Por último una vez ejecutado todo el proceso guiado siguiendo el documento guía, aparece un último documento, donde se plantean ejercicios para completar, una serie de cuestiones y la comprobación de si ha surtido efecto el aprendizaje, además de alguna cuestión a investigar por su cuenta. Esto se realizará de forma organizada definiendo los ejercicios para los distintos programas sin interactuar entre ellos, ya que al principio la dificultad de aprendizaje de cara el alumno es más alta para los usuarios menos experimentados.

Diseño e implementación de las unidades didácticas

3.2 – Práctica 2: Experimento FIR (Respuesta Impulsional Finita)

En esta práctica, entraremos de lleno en la teoría de filtrado y en el diseño de filtros.

Como ya sabemos existen dos grandes bloques de filtros: los filtros FIR (Respuesta Impulsional finita) y los filtros IIR (Respuesta Impulsional infinita). En este caso entraremos en materia con el primer grupo realizando un experimento completo de filtrado, donde se llevará una metodología ordenada y lógica para el desarrollo de cada uno de los procesos.

Los objetivos a conseguir en esta práctica son los siguientes:

- Conocimientos generales sobre el filtrado FIR, tipos de filtros y cuáles son los métodos de diseño.
- Generación de filtros en Matlab mediante el método de Parks-McClellan.
- Como exportar el filtro desde Matlab y como introducirlo manualmente en Code Composer Studio.
- Utilización de nuestro DSP para realizar un filtrado con el filtro generado
- Análisis de los resultados obtenidos del filtrado.

Siguiendo la metodología de uso de la herramienta, encontramos tres documentos. El primer documento corresponde a información general sobre el filtrado FIR. El segundo es un documento guía que se seguirá en el transcurso del experimento dirigiendo al alumno paso a paso a través de todas las tareas a ejecutar durante el mismo. Por último un documento con ejercicios planteados donde se comprueba la efectividad del aprendizaje, y se plantean algunas cuestiones a investigar por parte del alumno para continuar con el aprendizaje.

Tras la lectura y estudio del primer documento, el alumno habrá aprendido o recuperado sus conocimientos sobre el filtrado FIR, sus características, los tipos de filtros que se pueden realizar, y un conocimiento general de los métodos de diseño existentes.

Cuando se disponga a llevar a cabo el experimento tendrá a disposición la guía pulsando sobre el siguiente apartado del experimento.

En primer lugar, aunque en el esquema de la práctica nos dé a elegir varias opciones de filtros, y entradas, en la guía nos obligará a escoger un tipo de filtro y una entrada de audio por defecto, de entre la batería de señales que posee la herramienta para pruebas.

Diseño e implementación de las unidades didácticas

Se ha hecho así debido a que en un primer lugar es mejor que todos los alumnos lleguen a los mismos resultados, ya que ha escogido un caso inicial con un resultado muy visual, donde al finalizar el experimento se podrá analizar lo sucedido de una manera muy clara y no llegar a resultados con poca interpretación. Este caso se podría dar si se selecciona inadecuadamente el filtro para la señal de entrada elegida.

Ilustración 12: Cuadro Bloque Filtrado FIR

Una vez llegados a este punto, y después de la selección del filtro en Matlab, y la exportación de sus coeficientes se procede a la selección e introducción de la entrada de audio a procesar. Es necesario explicar algunos detalles del funcionamiento del programa en Code Composer Studio, ya que en la guía simplemente se insta a hacerlo funcionar, aquí comentaremos su modo de funcionamiento.

Diseño e implementación de las unidades didácticas

Nuestro programa en CCS, constará de 5 archivos de código escritos para llevar a cabo el experimento. Aparecen tres archivos de cabecera, y dos archivos de cuerpo:

- firCoef.h
- fixedPointFir.h
- tistdtypes.h
- fixedPointBlockFir.c
- fixedpointBlockFirTest.c

Entre estos cabe destacar el primero **firCoef.h**, donde se encuentran los coeficientes del filtro que se usaran en el experimento dentro de la variable (firCoefFixedPoint) cuyos valores podremos modificar con los filtros que generemos.

En segundo lugar encontramos el último **fixedpointBlockFirTest.c** donde se encuentran la organización del programa donde se selecciona el tipo de archivo a introducir, las impresiones por pantalla y la generación de la señal de salida. Además duran el proceso se llamara al último archivo reseñable:

En el penúltimo archivo de la lista **fixedPointBlockFir.c** se encuentra la labor de filtrado como tal, la aplicación de los coeficientes a los valores de entrada y la escritura de los valores de salida en la variable interna antes de su escritura.

Tras esto, queda comentar el análisis de resultados que se realizara en el script de Matlab otorgado, por el cual simplemente siguiendo la guía se facilitará su uso y su entendimiento.

Por último, una vez ejecutado todo el proceso guiado siguiendo el documento guía, se encontrará un último documento, donde se plantean ejercicios para proseguir con el experimento y comprobar si ha surtido efecto el aprendizaje, y alguna cuestión a investigar por su cuenta. Esto se realizará de forma organizada, definiendo los ejercicios donde se generarán filtros distintos a los que se encuentran en los scripts otorgados.

Diseño e implementación de las unidades didácticas

3.3 – Práctica 3: Experimento IIR (Respuesta Impulsional Infinita)

En esta práctica se continuará con el segundo gran bloque de filtrado, que son los filtros IIR por lo que al completar esta unidad didáctica, el alumno ya habrá realizado filtrados reales con varios ejemplos de cada uno de los dos grandes bloques de filtros.

Los objetivos a conseguir en esta práctica son similares a los de la práctica anterior, pero en este caso orientados al filtrado IIR:

- Conocimientos generales sobre el filtrado IIR, tipos de filtros y cuáles son los métodos de diseño.
- Generación de filtros en Matlab con la Toolbox especifica de diseño de filtros FDATool siguiendo el criterio de diseño de filtros elípticos.
- Como exportar el filtro desde Matlab y como introducirlo aprovechando la compatibilidad de la Toolbox con Code Composer Studio.
- Utilización del DSP para realizar un filtrado con el filtro generado
- Análisis de los resultados obtenidos del filtrado.

Siguiendo la metodología usada en las dos prácticas anteriores, se encuentran tres documentos de la misma forma: el primero de información general sobre el filtrado IIR. El segundo un documento guía que será seguido por el alumno en el transcurso del experimento dirigiendo a través de todas las tareas a ejecutar durante el mismo, y por último, un documento con ejercicios planteados donde se comprueba la efectividad del aprendizaje, y se plantean algunas cuestiones a investigar por parte del alumno para continuar con el aprendizaje.

De forma similar al experimento anterior se realizará la lectura y estudio del primer documento, el alumno habrá aprendido o recuperado sus conocimientos sobre el filtrado IIR, sus características, los tipos de filtros que se pueden realizar, y un conocimiento general de los métodos de diseño existentes.

Cuando se disponga a llevar a cabo el experimento tendrá a su disposición la guía pulsando sobre el siguiente apartado del experimento.

A diferencia del experimento anterior, en esté solo la herramienta no nos dejará decidir desde ella misma que tipo de filtro usar, sino que pulsando en el siguiente botón, nos abrirá un script de Matlab, que ejecutándolo nos dará acceso a la Toolbox de diseño de filtros. Sera dentro de ella donde la guía, nos haga elegir entre las distintas sesiones el filtro por defecto para la primera vez que realizamos el experimento.

Diseño e implementación de las unidades didácticas

Se ha hecho así debido a que, en un primer lugar es mejor que todos los alumnos lleguen a los mismos resultados, ya que al ser preparado anteriormente un ejemplo inicial contaremos con un resultado muy visual, donde al finalizar el experimento podremos analizar lo sucedido de una manera muy clara y no llegar a resultados con poca interpretación, como podría darse el caso si se selecciona inadecuadamente un filtro y la señal de entrada a filtrar.

Ilustración 13: Bloque herramienta Filtrado IIR

Una vez llegados a este punto, y después de la selección del filtro en la Toolbox de Matlab, se procede a la exportación de sus coeficientes aprovechando la vinculación que tiene la herramienta con Code Composer Studio. Además de la selección e introducción de la entrada de audio a procesar, es necesario explicar algunos detalles del funcionamiento del programa incluido en el experimento IIR de Code Composer Studio, ya que en la guía simplemente se insta a hacerlo funcionar, aquí se comentará su modo de funcionamiento.

Diseño e implementación de las unidades didácticas

El programa en CCS, constara de 6 archivos de código escritos para llevar a cabo el experimento. Aparecen cuatro archivos de cabecera, y dos archivos de cuerpo

- cascadeIIR.h
- fdacoefsMATLAB.h
- tistdtypes.h
- tmwtypes.h
- fixPoint_cascadeIIRTest.c
- fixPoint_cascadetIIR.c

Cabe destacar en este caso 3 de los archivos:

El primero se trata de **fdacoefsMATLAB.h,** donde se encuentran los coeficientes del filtro a emplear en el experimento, en este caso a diferencia del experimento de filtrado FIR, este archivo se generará por medio de la herramienta de exportación que incluye la Toolbox de Matlab, evitando de esta forma la farragosa tarea de introducirlos a mano. Este documento tendrá los coeficientes del filtro guardados en el formato propicio para su tratamiento en CCS posterior, separados los numeradores y los denominadores del filtro en distintas variables.

El segundo archivo destacable es **fixPoint_cascadeIIRTest.c.** Este archivo contiene el cuerpo principal del programa, como la adquisición de la señal de entrada y los mensajes por línea de comandos para realizar el experimento y seleccionar las distintas opciones, además realizara la llamada al último archivo de código a comentar.

En último lugar, **fixPoint_cascadetIIR.c,** contendrá la labor de filtrado a realizar usando los coeficientes introducidos y la señal de entrada de audio para, posteriormente, escribir los resultados en una variable interna antes de generar el archivo de audio de salida.

Para la realización del análisis de resultados se realizará el script de Matlab otorgado, por el cual simplemente siguiendo la guía que se nos facilita, seguiremos los diferentes pasos que simplificarán su uso y su entendimiento.

Finalizado todo el proceso guiado y terminando el documento guía, se encuentra un último documento donde se plantean ejercicios para proseguir con el aprendizaje y comprobar si ha surtido efecto la realización del experimento en la comprensión de los filtros IIR, y se incluirá alguna cuestión a investigar por su cuenta.

Esto se realizará de forma organizada definiendo los ejercicios donde se generarán filtros distintos a los que encontramos en los scripts otorgados.

Diseño e implementación de las unidades didácticas

3.4 – Práctica 4: Comparación de filtros

En ésta última práctica se cambiará la filosofía de los experimentos anteriores, a continuación se mezclará los temas anteriores, el bloque de filtrado FIR y el bloque de filtrado IIR de tal manera que a diferencia de en los dos anteriores experimentos que se centraba solo en una forma de diseño de filtros, en FIR Parks-McClellan y en IIR la metodología de filtros elípticos y se daba la opción de realizar los experimentos con todos los tipos de filtros, ya fuesen: pasa-bajos, pasa-altos, pasa-banda o para-banda. En éste la metodología de trabajo será la contraria, se elegirá en este caso filtros pasa-bajos, y se realizará todas las metodologías principales para el diseño de filtros, con el fin de comparar después de realizar el filtrado de una señal con todos ellos, cual realizo la labor de filtrado de mejor manera, más eficiente…

De tal manera el alumno dejará a su disposición una herramienta donde realizar sus pruebas y saber siempre que tipo de filtro responderá mejor, dado que se podrán cambiar los tipos de filtros a comparar y la señal de entrada.

Para facilitar esta práctica, se ha dejado de lado el DSP, para en última instancia aprender a hacer simulaciones de filtrado, usando Matlab.

Los objetivos de la práctica son:

- Sacarle el máximo partido a la Toolbox de diseño de filtros de Matlab (FDATool), utilizando todos sus métodos de diseño de filtros principales.
- Aprender a realizar simulaciones de filtrado con Matlab.
- Aprender a realizar una comparativa gráfica entre filtros donde se puedan sacar conclusiones claras.
- Aumentar el dominio de las representaciones gráficas en Matlab.
- Dar capacidad de decisión al alumno para saber discernir que filtro es mejor en cada caso, realizando comparaciones e interpretando los resultados por si mismo.

A pesar del diferente enfoque de la práctica, la metodología de trabajo de esta unidad didáctica, será la misma.

Se seguirá la herramienta de arriba abajo, y esta contendrá tres documentos:

El primero será de información general sobre la práctica. Tras la lectura de este documento, el alumno sabrá en que se basa la tarea que se va a realizar, y tendrá conocimiento de que partes se compone este experimento, y su finalidad.

Diseño e implementación de las unidades didácticas

El segundo documento, será la guía, que de igual manera que en experimentos anteriores. Se realizará su lectura mientras se llevan a cabo las diferentes partes de la práctica.

Ilustración 14: Bloque herramienta Comparación de filtros

En este caso, en el experimento de comparativa de filtros, está preparado con diez filtros a comparar, cinco correspondientes a filtrado FIR y cinco al filtrado IIR.

Las sesiones están generadas mediante la herramienta FDATool. De esta manera se maximizará el partido que le sacamos a esta Toolbox dado que habremos utilizado prácticamente todos los tipos de filtros que nos da opción de diseñar.

Se guardan los diseños de los filtros pasa-bajos, todos realizados con la misma filosofía, una frecuencia de paso de 2KHz, y una frecuencia de stop de 2,5 KHZ.

La frecuencia de muestreo de todos los filtros será de 24 KHz.

Tras exportar los filtros al workspace de Matlab, la guía nos instará en abrir el script de comparación de filtros.

Diseño e implementación de las unidades didácticas

Éste está preparado con distintas secciones, donde al ejecutar la primera se generará una señal a filtrar. Se ha dejado como comentario unas líneas de código que introducen ruido a la señal generada, haciendo de esta manera el experimento más realista, pero en un primer momento en la guía se desaconseja su uso, ya que cuanto más simples sean los resultados en un primer momento, la interpretación de los mismos será más sencilla.

Una vez ejecutado la comparativa se verán dos gráficas, una que contendrá los filtros FIR y otra con los filtros IIR. Se han separado por el hecho de que representar diez funciones en una gráfica es caótico y no se visualiza de forma clara los resultados.

De esta manera al tener que realizar una separación se decidió hacerla por el tipo de filtro.

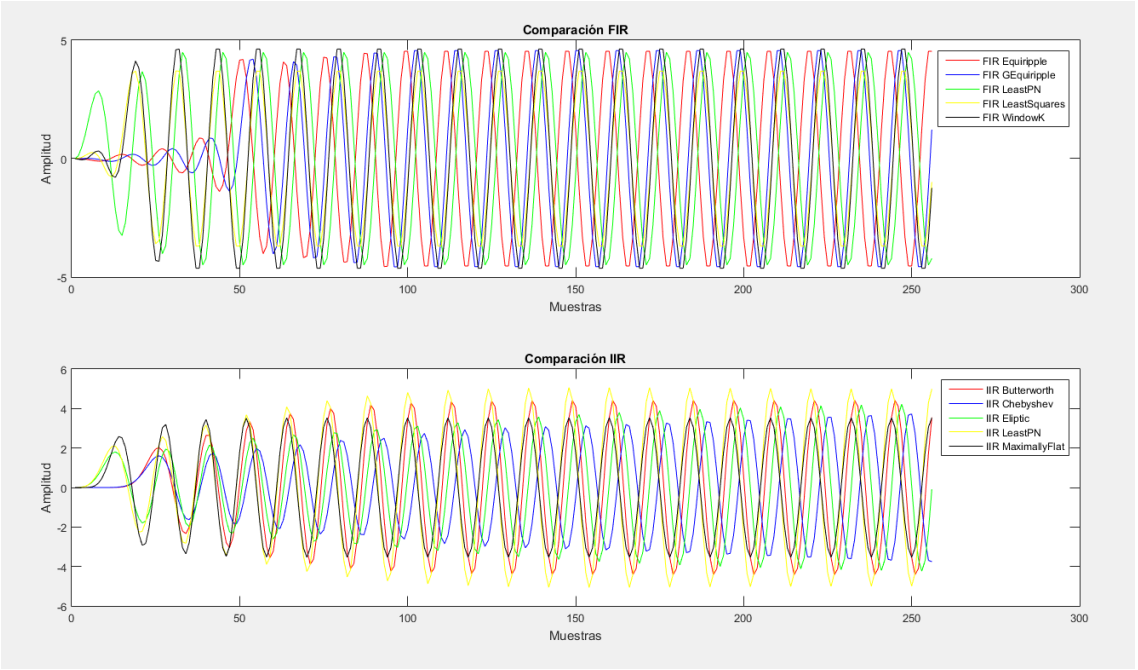


Ilustración 15: Doble gráfica

En el tercer documento que se encuentra, corresponde a los ejercicios planteados una vez terminado el experimento en cuestión. En este caso, los ejercicios realizarán muy pocas cuestiones sobre el experimento. Por otra parte, orientarán al alumno a realizar por completo el experimento tres veces más, modificando las sesiones otorgadas de los filtros para realizar la comparativa de filtros pasa-altos, pasa-banda y para-banda.

Diseño e implementación de las unidades didácticas

No será un trabajo arduo, dado que modificar las sesiones una vez generadas para los filtros no tiene gran dificultad. El script de la comparación con las representaciones puede reutilizarse con la única modificación interesante a realizar que sería la generación de la señal de entrada, para acercarlas en cada caso a los límites del filtrado. Esto permitirá ver en cada situación, como se comportan los filtros, mostrando directamente la representación de sus resultados.

Diseño e implementación de las unidades didácticas

3.5 – Implementación de la Interfaz

Una vez terminadas las unidades didácticas con sus guías y ejercicios, como se comentó anteriormente en la introducción, una vez se hayan realizado las practicas realizadas, y todos sus recursos, se creará una interfaz que envuelva el programa, y nos facilite el uso de los experimentos.

Objetivos de la interfaz:

- Generar una ventana que organice los distintos experimentos de tal manera que su seguimiento sea más intuitivo.
- Apertura sucesiva de las aplicaciones que necesitemos en cada caso a través de los botones incluidos en la ventana. Este proceso hará las veces de lanzador.
- Dar al proceso de aprendizaje una visión atractiva del mismo, para que el usuario se sienta más cómodo usando la herramienta.

Para realizar la interfaz, en primer lugar se realizó un esquema, con los bloques principales de la aplicación como las diferentes prácticas a realizar en la herramienta.

Por lo que dispondría de cuatro bloques, uno para cada experimento. Estos cuadros estarían organizados de izquierda a derecha, según el orden de realización, siendo el primero el bloque de introducción en la izquierda, y el ultimo el relativo a la comparación de filtros el más a la derecha.

Dentro de estos bloques, se propuso que los botones que contendrían, recorrerían el contenido a realizar de arriba abajo, es por ello que en primer lugar se colocaron los documentos de información general sobre el experimento en cuestión, junto con la guía para su seguimiento, que dirá en todo momento que botón se debe pulsar y cuando se debe o no cerrar cada aplicación. Y en la parte más baja del cuadro de cada experimento, se encuentran los ejercicios.

Nótese que cuando se realizan los experimentos FIR y IIR, no se utilizan todos los archivos preparados en cada bloque.

Es por ello que la aplicación tiene una interfaz recursiva, está preparada para que una vez se realice el experimento por defecto, el alumno pueda seguir realizando otros experimentos de manera individual cambiando algunas de las variables como por ejemplo el tipo de filtro o la señal de entrada a procesar.

De esta manera la herramienta proporciona muchas más horas de aprendizaje autónomo de las que se realizarán en el laboratorio de la asignatura de procesado digital de señal.

Diseño e implementación de las unidades didácticas

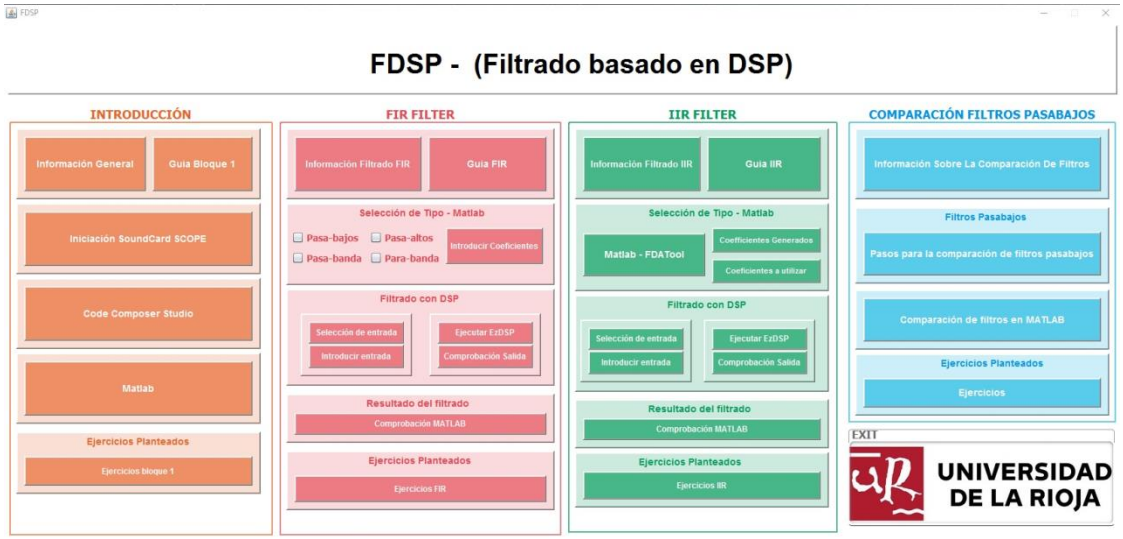


Ilustración 16: Interfaz herramienta - Java

Una vez terminado el esquema, se utilizó Netbeans para su realización, introduciendo con la herramienta de diseño los siguientes botones, y creando la estructura vista en la ilustración 16.

Se introdujo un botón de salida en el botón de Universidad de la Rioja para dar otra posibilidad de cierre de la aplicación.

Tras generar el esquema, queda realizar la apertura de los documentos o las aplicaciones concretas en función de que es lo que se quiera abrir.

Para ello si utiliza un manejador de eventos, que abrirá el archivo vinculado a la dirección introducida con la aplicación Windows que tengamos asignada por defecto.

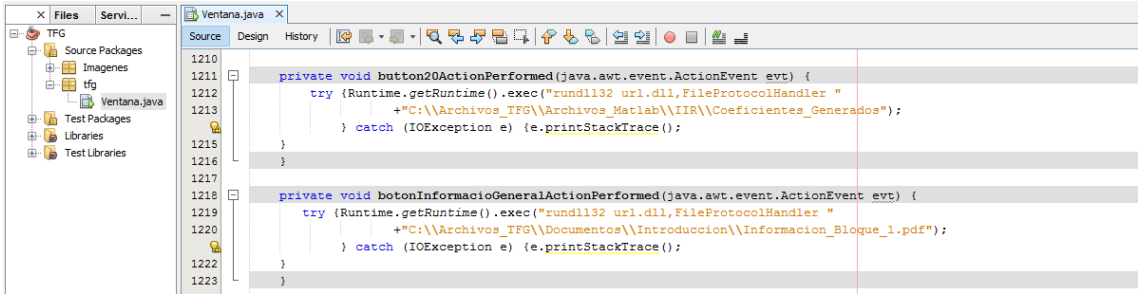


Ilustración 17: Manejador de eventos

Se realizará la llamada al manejador de eventos y este será quien leyendo las extensiones de archivo, buscará la aplicación Windows definida por defecto para abrir el archivo en cuestión.

En el caso de no tener extensión de archivo, entenderá que se trata de una carpeta, y abrirá la dirección con el explorador de Windows.

Y en el caso de encontrar una extensión de archivo no reconocida, abrirá el menú de opciones de Windows para seleccionar la aplicación con la que abrir el archivo contenido en la ruta de dirección.

4 - Conclusiones:

Como se ha comentado en los puntos anteriores, este TFG tiene los objetivos muy marcados, y se han ido desarrollando durante las distintas unidades didácticas, puntualizando en los puntos más importantes en las guías y cercando el ámbito donde el alumno debe sacar sus conclusiones.

El alumno puede de una forma casi inconsciente obtener la consecución de muchos de los objetivos durante la realización de las prácticas. Como ejemplo, puede ser el implementar un entorno de trabajo basado en DSP, donde ellos mismos en un futuro puedan recurrir a la herramienta modificando sus archivos para adaptarlos a las dudas que les puedan surgir más adelante y solucionarlas por si mismos dentro del ámbito del filtrado digital de señal.

Otro de los objetivos que pasan más desapercibidos por parte del usuario que utiliza la herramienta, es el hecho de que solo uno de los experimentos que se realiza es una simulación, todos los demás son reales. Además, no siempre siguen el comportamiento esperado, ya que aparecen muchos otros factores que varían el funcionamiento del experimento.

La herramienta de trabajo incluye elementos que determinan nuestros resultados, como el hecho de introducir señales generadas por la tarjeta de sonido del PC: Esta señal es filtrada por el filtro generado en el DSP, añadiendo un punto de incertidumbre, por lo que en casos concretos podrán aparecer comportamientos erráticos e injustificados.

Es por eso, que al generar los experimentos se ha tenido especial cuidado y se han realizado repetidas pruebas, para poder sacar las conclusiones adecuadas.

El camino realizado hasta llegar a la consecución de una herramienta fiable y práctica no ha estado exento de obstáculos, pero ha servido para estar más satisfecho por los resultados obtenidos.

Otra de los objetivos más interesantes de este proyecto, es la adquisición de la metodología de filtrado. Es decir, definir el tipo de filtro, su generación, la adquisición de los coeficientes, su exportación, la generación de una señal de entrada, la construcción de un programa de filtrado real, la ejecución en un DSP real, y la recepción de los resultados que se analizarán para sacar conclusiones.

Todos estos pasos, se realizan en diferentes programas, que habrá que compatibilizar, porque de qué sirve un programa de generación de filtros, si luego el método para

Conclusiones

exportarlos no sirve en el programa que realizará el filtrado. Por eso se ha tenido muy en cuenta la compatibilidad entre herramientas de tal manera que se pueda seguir el proceso de filtrado, sin perder la visión objetiva de lo que se está haciendo y con qué fin.

A efectos prácticos al realizar este proyecto se ha logrado alcanzar de manera satisfactoria todos los objetivos principales para los que en su día, el director de proyecto y yo lo definimos.

Como se mencionó en la introducción de este documento el uso de esta herramienta no está limitado al uso en el aula para estudiantes de ingeniería, sino que los contenidos están adaptados tanto a un usuario con conocimiento en la materia, como al desconocedor de la teoría del filtrado digital.

Sí que es cierto que para una persona desconocedora de la materia, quizás debiese buscar más información teórica, que la dada por la herramienta durante el desarrollo de los experimentos, para contrastar y asentar de una manera más sólida los conocimientos teóricos, pero el objetivo principal de esos documentos es refrescar los conocimientos de aquellos que ya lo han estudiado con anterioridad.

Cabe mencionar también en último lugar que será un orgullo para mí que este proyecto vea la luz en su uso en esta Universidad y que sirva para la formación de sus futuros ingenieros.

5 - Anexos:

Contaremos con cinco anexos, tantos como apartados encontramos en el apartado 3 de este documento, y que además se centrarán en los mismos bloques.

De esta manera, cada uno de los anexos contendrá todo el contenido a mostrar por parte de la aplicación, como son todos los documentos mencionados en el apartado 3 de los que disponía la herramienta en cada una de las unidades didácticas, además del código con el que se crearon las distintas aplicaciones.

- Anexo 1: Introducción a la herramienta.
- Anexo 2: Filtrado FIR.
- Anexo 3: Filtrado IIR.
- Anexo 4: Comparación de filtros.
- Anexo 5: Desarrollo de la interfaz.

Anexos

Anexo 1:

Filtrado Digital de Señal Con DSP

Introducción General

Iñigo Rodríguez Martínez

Tutor: Antonio Moisés Zorzano Martínez

31/01/2017

Procesador de señal a emplear: **eZdsp TMS320VC5505 USB Stick**

UNIVERSIDAD DE LA RIOJA

A lo largo de este documento se describen los contenidos de una serie de unidades didácticas a desarrollar por el alumno, para el aprendizaje a través de la práctica de la asignatura de procesado digital con experimentos reales.

Anexos

Contenido

Anexos

Descripción de los elementos

Software

En el programa principal (la aplicación Java: **"Launcher"**), encontramos las unidades didácticas a tratar en cuatro bloques:

El primer bloque, se trata de un bloque de introducción a la herramienta, donde trabajaremos de forma individual, todos los programas que se utilizarán a lo largo del proceso de aprendizaje como: ***(Code Composer Studio, Matlab, SoundCard Scope)***, siendo totalmente guiado, se seguirán las instrucciones contenidas en este documento para la resolución de distintos ejercicios con los que trataremos de adquirir habilidades básicas que emplearemos de forma directa o indirecta en los siguientes apartados del programa.

SoundCard Scope: Se trata de un programa de Licencia libre para uso educativo, que nos da la posibilidad de utilizar un osciloscopio digital donde se podrá: analizar señales de audio de distintas formas, visualización de la onda, análisis de su espectro frecuencial…

También permitirá, la generación de señales de audio mediante la tarjeta de sonido de nuestro ordenador, las cuales podrán generar y modificar para escucharlas durante la generación, o crear archivos de audio que se puedan procesar más adelante.

Recibe los datos desde la tarjeta de sonido con una frecuencia de muestreo de 44.1 KHz y una resolución de 16 bits, pudiéndose seleccionar el origen de los datos desde el mezclador de Windows. El rango de frecuencias va a depender de la tarjeta de sonido empleada, un rango de 20Hz a 20KHz no debería suponer un problema con cualquier tarjeta moderna.

CCS: un programa desarrollado por Texas Instruments, cuya licencia está ligada a la compra de la tarjeta: ***(eZdsp TMS320VC5505***) que se empleará durante los ejercicios de procesado digital en los distintos bloques.

Contiene un conjunto de herramientas para desarrollar y depurar programas destinados a sistemas embebidos, con un compilador de C/C++, un editor, herramientas de depuración…

Matlab: se trata de una plataforma optimizada para solucionar problemas de Ingeniería y científicos. Contiene una vasta librería de **toolboxes** preinstaladas que permiten empezar a trabajar inmediatamente con algoritmos esenciales para su dominio. En nuestro caso se dará un especial énfasis a la **FDATool**, una Toolbox destinada al diseño de filtros digitales. El uso de esta herramienta otorga una ventaja ya que todas las herramientas están probadas y diseñadas rigurosamente para trabajar juntas.

Anexos

Los filtros digitales se pueden clasificar en dos grandes grupos: aquellos que presentan una respuesta al impulso de duración infinita (IIR) y, por el contrario, los sistemas FIR o de respuesta al impulso finita.

El segundo bloque, está dirigido al filtrado FIR *(Finite Impulse Response)*, en este bloque encontraremos una introducción al filtrado FIR, con un repaso a la teoría necesaria para poder seguir y entender los siguientes experimentos que realizarán.

Recorriendo este bloque, se realiza un filtrado completo Real no se trata por tanto de una simulación**.**

Se utilizará **código** en script de **Matlab,** para la generación de un filtro, pasando por la elección del tipo de filtro (pasa-bajos, pasa-altos…) siguiendo el criterio de **Parks-McClellan**, diseñándolo según nuestras especificaciones y calculando sus coeficientes. Esto se verá suavizado con ayuda de la guía incluida que el alumno podrá seguir paso a paso, para facilitar el proceso.

El experimento incluido, realizará un procesado digital de una señal de audio, donde podremos escuchar la señal de entrada, realizaremos el procesado de la señal con ayuda de CCS **(Code Composer Studio)**, para la que necesitaremos nuestra tarjeta ezDSP, utilizaremos los coeficientes del filtro generados en el apartado anterior, y realizaremos el procesado.

Tras esto podremos analizar el resultado obtenido en una señal de audio resultado, procesada por nuestro ezDSP.

Para poder seguir trabajando con la herramienta se plantean en el último apartado de este bloque, una serie de **ejercicios** a realizar por el alumno de forma independiente, para que puedan seguir aprendiendo realizando otros ejemplos de procesado digital mediante filtros FIR, como puede ser: cambio de tipo y especificaciones del filtro, cambio de la entrada de audio a procesar para ver las diferencias en las salidas, generación de señales de audio a procesar…

Se podrá recorrer el segundo bloque tantas veces como se quiera dado que se ha preparado una batería de filtros con cada uno de los tipos que podremos modificar y ajustar los experimentos de filtrado, otorgando así la oportunidad de aprender analizando los resultados ***reales*** *de cada uno de los filtros. Además se podrán ajustar los parámetros para encontrar fallos típicos en el procesado como por ejemplo la insuficiencia en la frecuencia de muestreo que nos dará lugar a una señal errónea a la que deberíamos obtener (Aliasing).*

Anexos

El tercer bloque, está dirigido al filtrado IIR (Infinite Impulse Response), utilizando así los dos grandes grupo de filtrado. También encontraremos un repaso a la teoría necesaria para facilitar el seguimiento de los siguientes experimentos a realizar.

Al igual que le bloque anterior, los experimentos son reales.

En este caso, cambiaremos la forma de diseñar los filtros, para presentar la herramienta que nos ofrece **Matlab** para facilitar el proceso. Utilizaremos la Toolbox destinada a la generación de filtros y cálculo de coeficientes que nos ofrece: FDATool ("Filter Design & Analysis Tool").

El programa ofrecerá un filtro de cada tipo (pasa-bajos, pasa-altos…) fijando el criterio de diseño como filtros **elípticos**, con todos los parámetros introducidos para que el alumno pueda ver como se realiza la configuración de una forma sencilla, para facilitar un primer uso de la herramienta.

Se extraerán los coeficientes para utilizarlos en el siguiente paso.

El experimento incluido, realizará al igual que en el segundo bloque un filtrado de un archivo de audio, que podremos cambiar si lo deseamos, e introduciremos los coeficientes del filtro generados por la FDATool, se realizará un procesado de la señal mediante CCS (Code Composer Studio), para la que necesitaremos nuestra tarjeta ezDSP,

Tras esto podremos analizar el resultado obtenido en una señal de audio resultado, procesada por nuestro ezDSP.

Para poder seguir trabajando con la herramienta se plantean en el último apartado de este bloque, una serie de **ejercicios** a realizar por el alumno de forma independiente, para que puedan seguir aprendiendo realizando otros ejemplos de procesado digital mediante filtros IIR, como puede ser: cambio de tipo y especificaciones del filtro, cambio de la entrada de audio a procesar para ver las diferencias en las salidas y poder analizarlas.

*Al igual que con el segundo bloque podremos recorrer este tantas veces como queramos dado que se ha preparado una batería de filtros con cada uno de los tipos que podremos modificar y ajustar los experimentos de filtrado, otorgando así la oportunidad de aprender analizando los resultados **reales** de cada uno de los filtros. Y realizar un análisis detallado de sus resultados sacando las conclusiones correspondientes.*

Anexos

El cuarto bloque, y último, cambiará por completo el enfoque para seguir aprendiendo, anteriormente, realizando diferentes tipos de filtros sobre criterios de generación fijados para comprender la realización del procesado y sus resultados. También dejará de lado los experimentos reales para acortar la comparativa, por lo que todo el bloque se desarrollara en **Matlab** y su toolbox de generación de filtros **FDATool**.

En este bloque se tratará de comprobar la eficiencia de los distintos filtros fijando el tipo de filtro **Pasa-bajos**, y probando los diferentes criterios de diseño de filtros, tanto IIR como FIR.

De esta manera se podrá simular el filtrado de una señal generada por el usuario mediante un script de Matlab, y pudiendo visualizarla.

Se realizará el filtrado de esta señal con los diferentes filtros pasa-bajos generados, y se comparará los diferentes gráficos resultantes del filtrado para comprobar la eficiencia de los mismos y saber realizar una toma de decisiones en cada caso sabiendo que filtro es el más conveniente utilizar en función de nuestras necesidades.

Tras esto el script se podrá **modificar fácilmente** para poder realizar otras comparativas de filtrado para otros tipos de filtro, pasa-altos, pasa-banda, para-banda…

Por último se plantea una serie de ejercicios para comprobar los conocimientos adquiridos y la capacidad de **elección** entre los diferentes filtros.

Anexos

Hardware

TMS320VC5505 eZDSP USB Stick

El C5505 es un procesador de señales digitales (DSP) que forma parte de la familia C55 de DSPs en coma fija de Texas Instruments (TI) diseñado especialmente para el desarrollo de aplicaciones de bajo consumo.

Las siglas DSP (Digital Signal Processor/procesador digital de señales) hacen referencia a microprocesadores que han sido diseñados para poseer un juego de instrucciones, un hardware y un software optimizados para aplicaciones en las que se requieren operaciones numéricas de muy alta velocidad.

Con la capacidad de manipular señales digitales en tiempo real mediante secuencias de instrucciones iterativas a muy alta velocidad. La diferencia principal entre un DSP y un microprocesador es que el DSP tiene unas especificaciones diseñadas para soportar tareas de altas prestaciones, con alto número de repeticiones y numéricamente intensas.

Los DSP suelen usarse para medir, filtrar y/o comprimir señales analógicas (ej. filtrar o convertir audio en diferentes formatos). Realizando tratamiento de señales en donde convierte las señales analógicas en un flujo de muestras digitales a través de un ADC (Analog/Digital Converter).

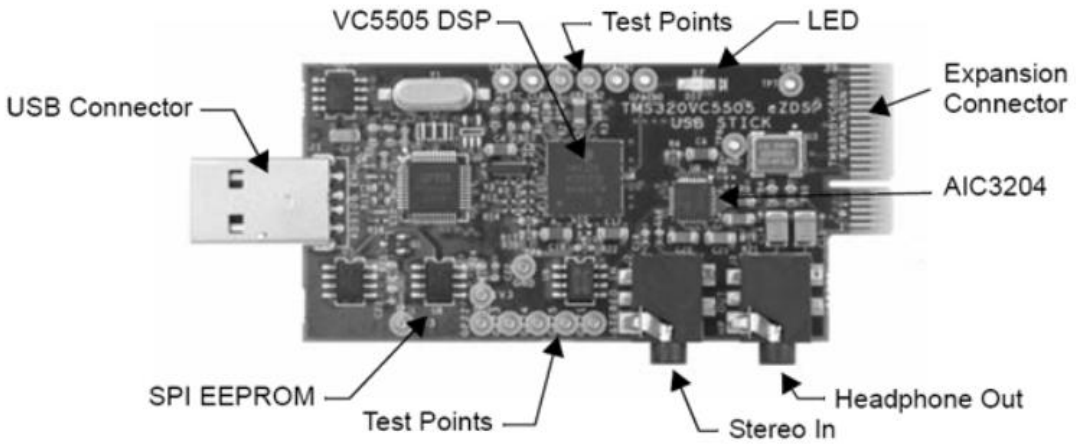


Ilustración 18: C5505 elementos

Anexos

Filtrado Digital de Señal Con DSP

Guía Bloque 1

Iñigo Rodríguez Martínez

Tutor: Antonio Moisés Zorzano Martínez

31/01/2017

Procesador de señal a emplear: **eZdsp TMS320VC5505 USB Stick**

UNIVERSIDAD DE LA RIOJA

A lo largo de este documento se desarrollan los pasos a seguir en los diferentes apartados del primer bloque para la iniciación del alumno en los diversos programas que se utilizarán durante el desarrollo de las distintas unidades didácticas.

Anexos

Contenido:

Anexos

Introducción:

Tras la lectura del documento de introducción general, en esta guía se encontrarán los pasos a seguir para la utilización de todos los programas que se utilizaran en las siguientes sesiones.

El alumno en un primer momento deberá simplemente seguir los pasos descritos y limitarse a adquirir los conceptos y la metodología que se describe a continuación.

Si no se siguen los pasos se podrán dar errores de compilación u otros resultados finales que nos son los que busca presentar esta guía, por lo que se realizarán las pruebas una vez terminado el seguimiento de los pasos.

SoundCard Scope (SCS):

Al abrir SCS por primera vez, se comprobará que nada más abrirlo, tiene una forma similar a un osciloscopio digital. Cuenta con dos canales, donde como en cualquier osciloscopio se podrá ajustar la amplitud, el tiempo y los disparos.

El programa cuenta con 6 módulos, las 6 pestañas que se pueden ver en la parte superior que permitirán el movimiento entre las diferentes utilidades de la herramienta.

En un primer momento se encuentra la Generación de señales, por lo que se utilizará la cuarta pestaña dedicada a la generación y la quita “Extras” dedicada a la grabación.

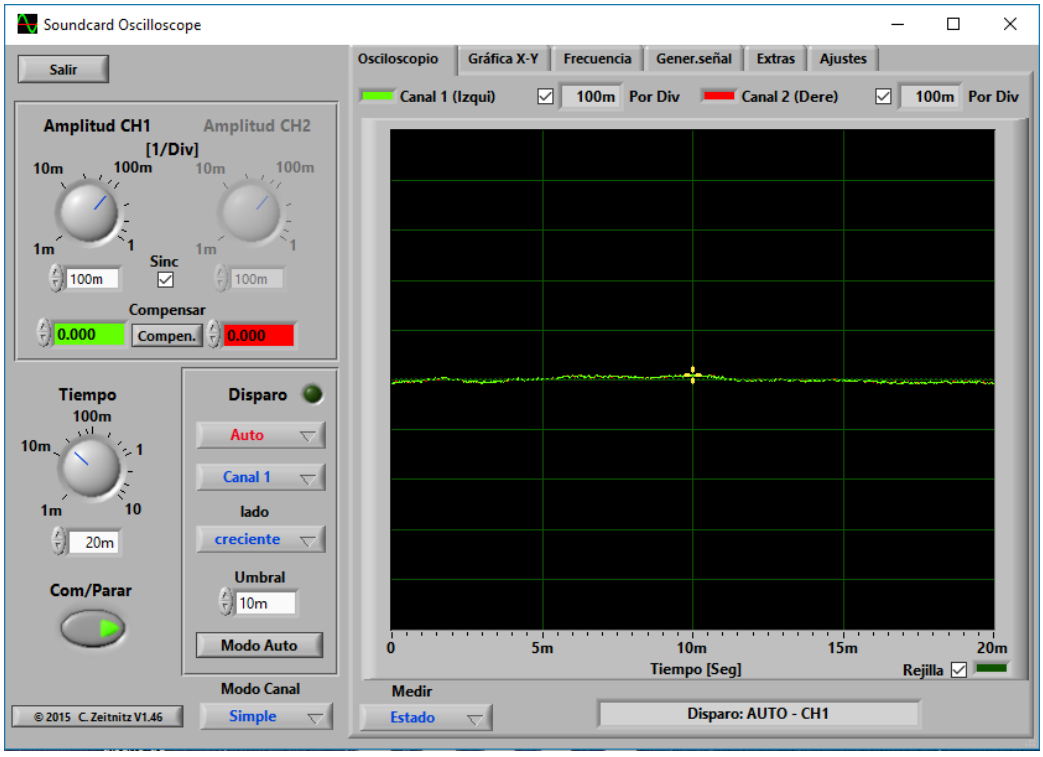


Ilustración 19: SoundCard SCOPE

Anexos

1. Aprenderemos a generar señales de entrada mediante la tarjeta de sonido del pc. Guardaremos estas señales en forma de archivo (.wav) para poder utilizarlas más adelante en nuestros experimentos.

2. En primer lugar iremos a la pestaña de "Extras" donde prepararemos la grabación del archivo a generar. En primer lugar activaremos la opción en el desplegable de guardar cuando le demos al botón digital. Quitaremos la opción de disparo automático, y le pondremos a la señal la longitud deseada. Se advierte de que longitudes largas, darán lugar a procesamientos más largos de las señales de entrada. Aunque para realizar pruebas será más didáctico realizar grabaciones más largas.
   Por último la ruta de destino de la grabación donde queréis tener vuestras señales.

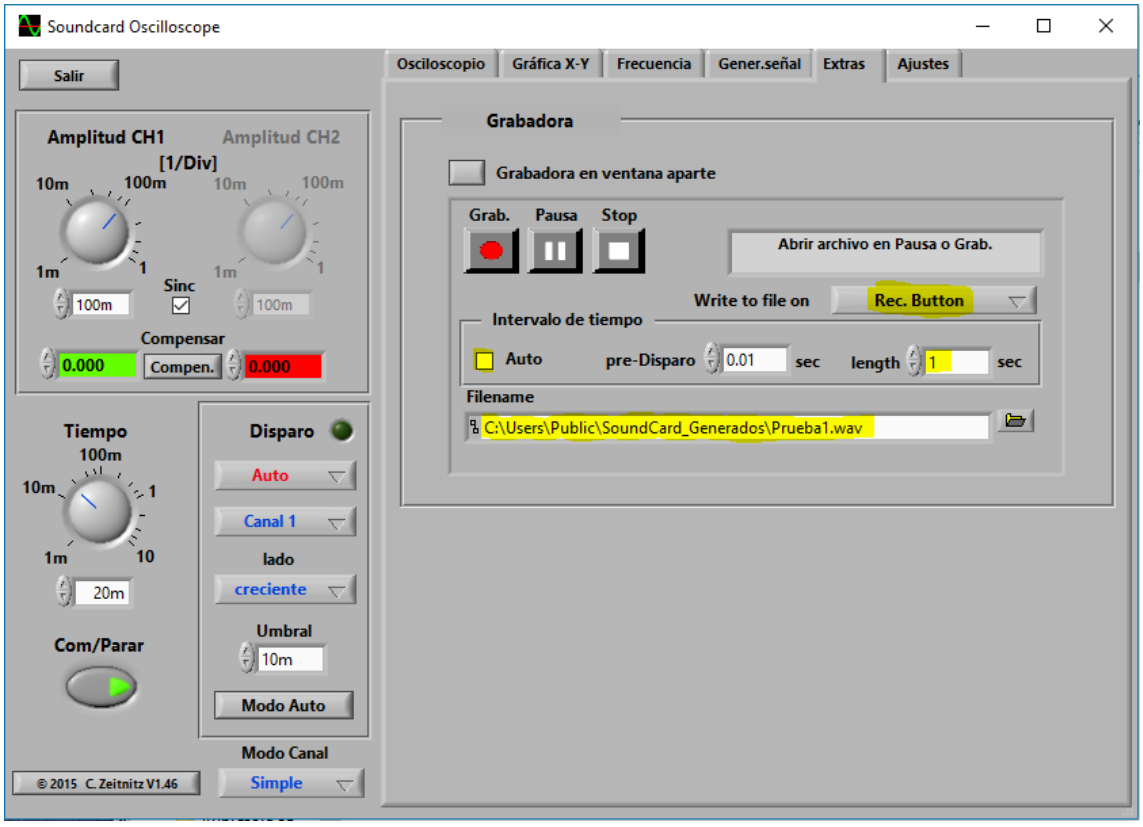


Ilustración 20: Grabación SCS

3. Una vez preparada la grabación vamos a la pestaña de generación de señal, posee dos canales, por lo cual nos permitirá generar dos señales de forma independiente por cada uno de ellos., nos centraremos en el canal 1, donde primeramente dejaremos la forma de onda senoidal por defecto, la amplitud de la señal de 0.5 y la frecuencia en 100 Hz para que el sonido no sea muy molesto para los oídos, en la sección de más abajo llamada ""Send to Scope" enviaremos la señal generada al canal 1, tras esto la activaremos con el botón superior al lado del nombre del Canal1.

Anexos

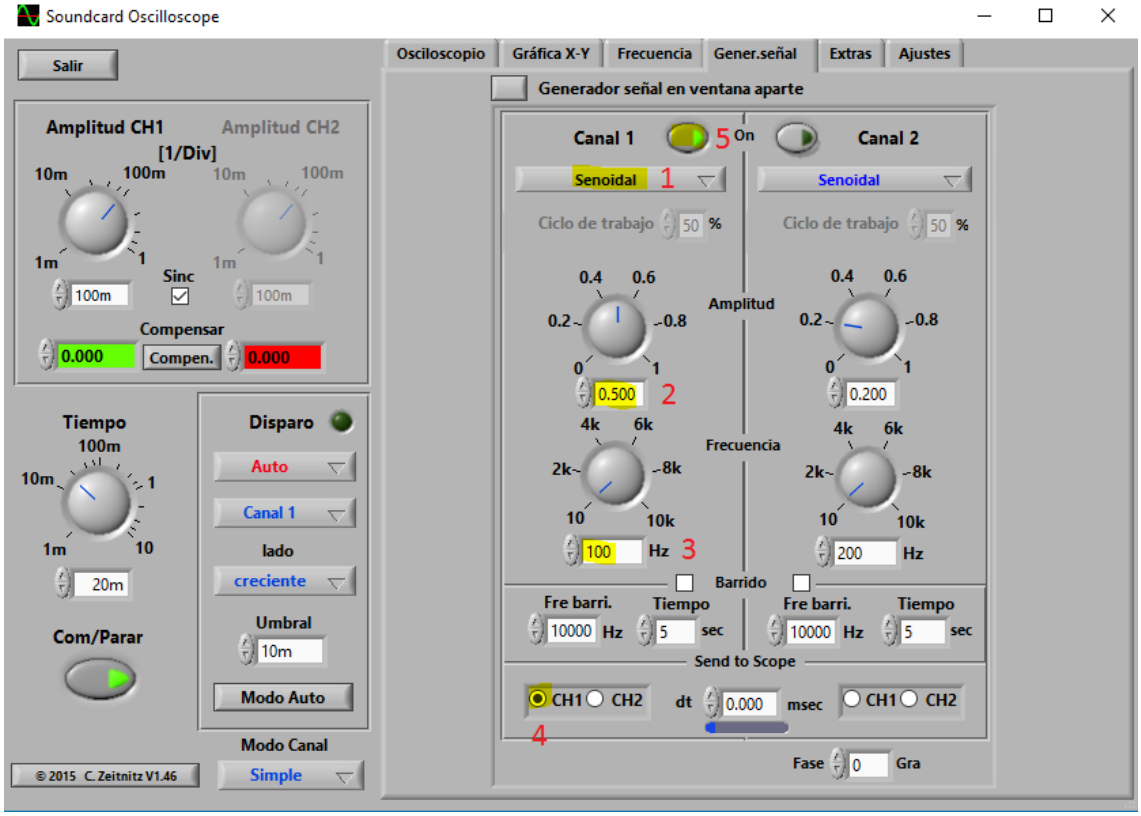


Ilustración 21: SCS Generador de señales

4. A continuación ya estamos escuchando la señal, procederemos ahora a guardarla, por lo que volvemos a la ventana de "Extras" donde tenemos todo configurado del punto 2, y procedemos a darle a "Grab."(botón del círculo rojo), que pondrá el modo grabación, tras un segundo clic en lo botón comenzará a grabar. Sabemos cuándo está grabando ya que nos lo indica con texto "Grabar", cuando desaparece este texto, debemos darle a "STOP" para que realice el guardado del archivo.
5. Ahora volveremos a la pestaña de generación de señal y apagaremos la generación para evitar el ruido en el aula.
6. En último lugar escucharemos el archivo creado guardado en el directorio de destino para corroborar que se ha grabado correctamente.

Si encontramos algún problema a la hora de grabar, hay un problema típico que se da cuando efectuamos la grabación, en la última pestaña de SCS, "Ajustes" nos dejará seleccionar el dispositivo de entrada y de salida. El de salida han de ser lo altavoces, y el de entrada, tenemos que seleccionar: loopback: "altavoces", si no, nos permite seleccionarlo o nos cambia la decisión, es porque tenemos un micro conectado, deberemos deshabilitarlo para que nos deje poner el loopback de los altavoces como dispositivo de entrada.

Anexos

CCS (Code Composer Studio):

Introducción

En primer lugar en esta práctica realizaremos, como casi siempre suele hacerse en la mayoría de los dispositivos, un "Hola Mundo", donde aprenderemos a **crear un programa en CCS** y **transferirlo a la tarjeta.**

Pasos para la realización del ejercicio:

1. En primer lugar conectaremos nuestra tarjeta **(TMS320VC5505 eZdsp USB Stick)** al pc. Arrancamos la aplicación CCS v4, cuando abrimos CCS por primera vez, el programa debería solicitar al usuario el área de trabajo **(workspace)** a utilizar, que es la ruta física donde van a colgar los proyectos. A continuación vemos la ventana:

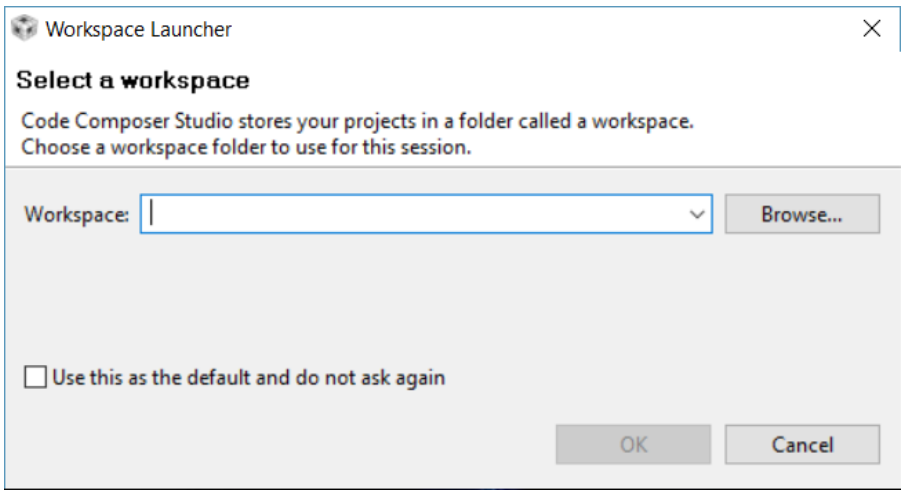


Ilustración 22: workspace CCS

(Se recomienda no marcar la casilla "Use this as default and do not ask again" para que dicha ventana aparezca en cada sesión de laboratorio aunque se puede llegar a este lugar navegando por las opciones del CCS).
En esta ocasión llamaremos a la carpeta de la práctica: Práctica 1.

El entorno generalmente tendrá la estructura de la imagen inferior, teniendo varias secciones definidas, que nos permitirán interactuar con el programa.

Anexos

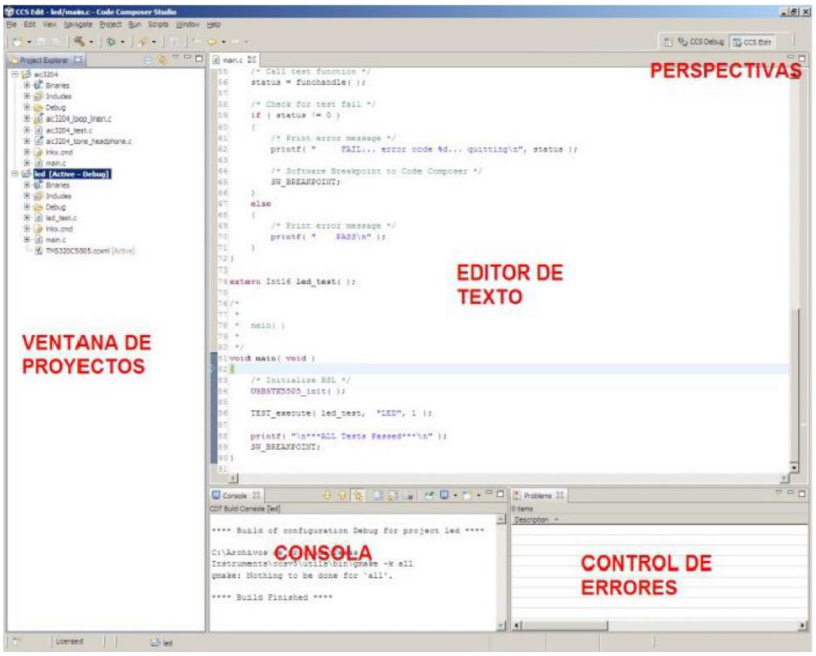


Ilustración 23: CCS elementos

2. Crearemos un proyecto nuevo:

File->New->CCS Project: Aquí se nos abrirá una ventana donde elegiremos el nombre del proyecto: "Practica 1" y su localización, que será la definida como workspace.

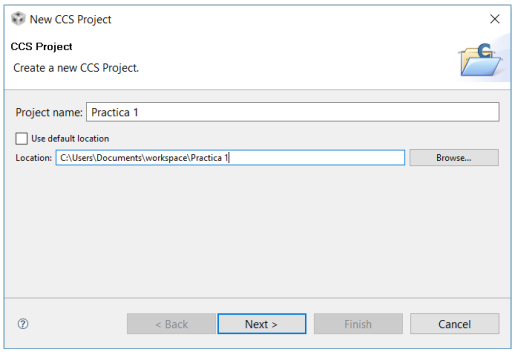


Ilustración 24: Nuevo proyecto CCS

A continuación, elegiremos la familia del dispositivo a emplear:

Anexos

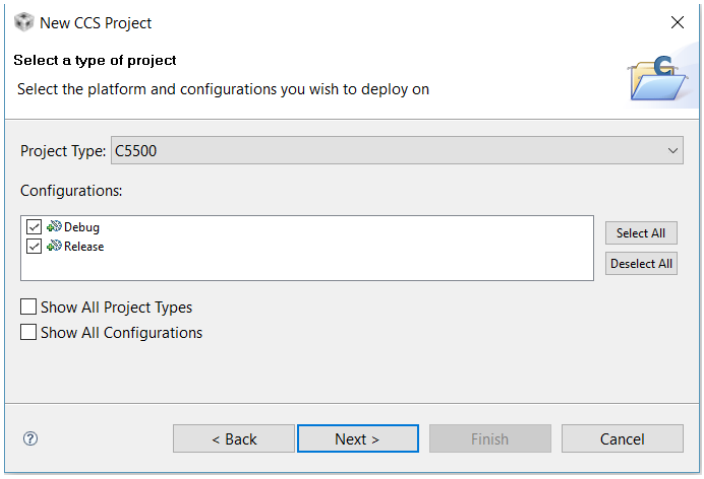


Ilustración 25: CCS configuración proyecto

Continuaremos, sin añadir ningún archivo al proyecto, y terminaremos la configuración seleccionando:

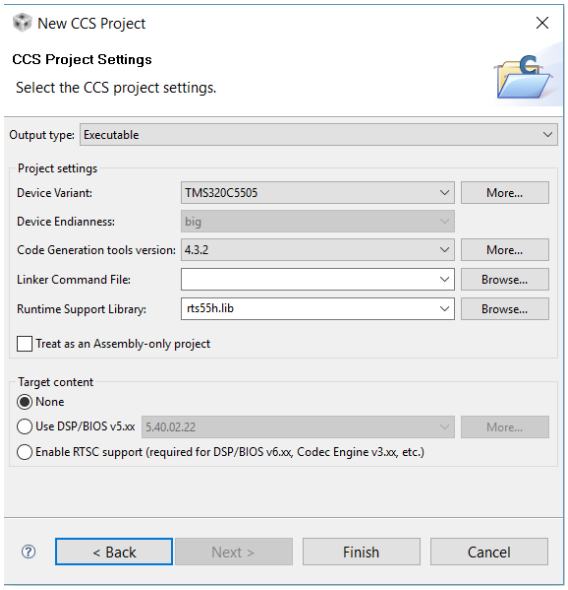


Ilustración 26: CCS configuración proyecto 2

3. Añadimos la sección de código al proyecto:

File->New->Source File: Le daremos el nombre: **main.c**

Anexos

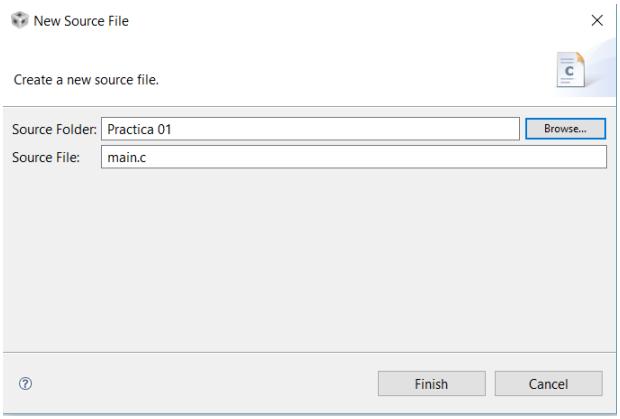


Ilustración 27: CCS configuración proyecto 3

Y escribiremos el programa escrito a continuación:

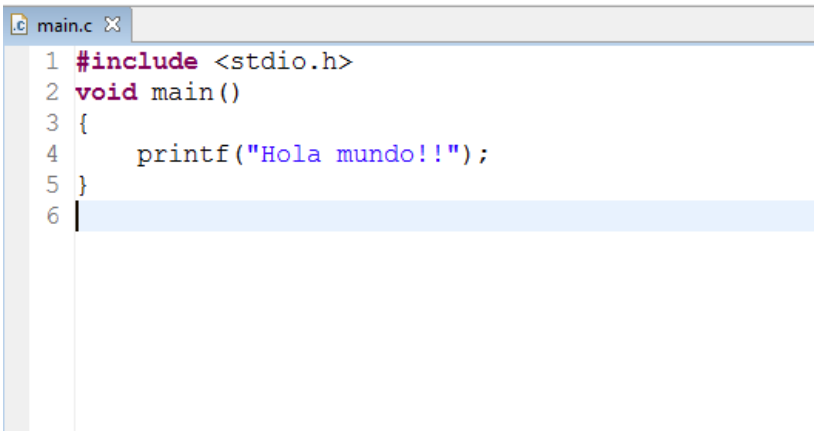


Ilustración 28: CCS programa

A continuación **guardamos**, y en la ventana de proyecto clicamos con el botón derecho en la carpeta de proyecto, y seleccionamos **propiedades**.

A continuación se nos desplegará un menú, donde en el apartado: **C/C++ Build**

Anexos

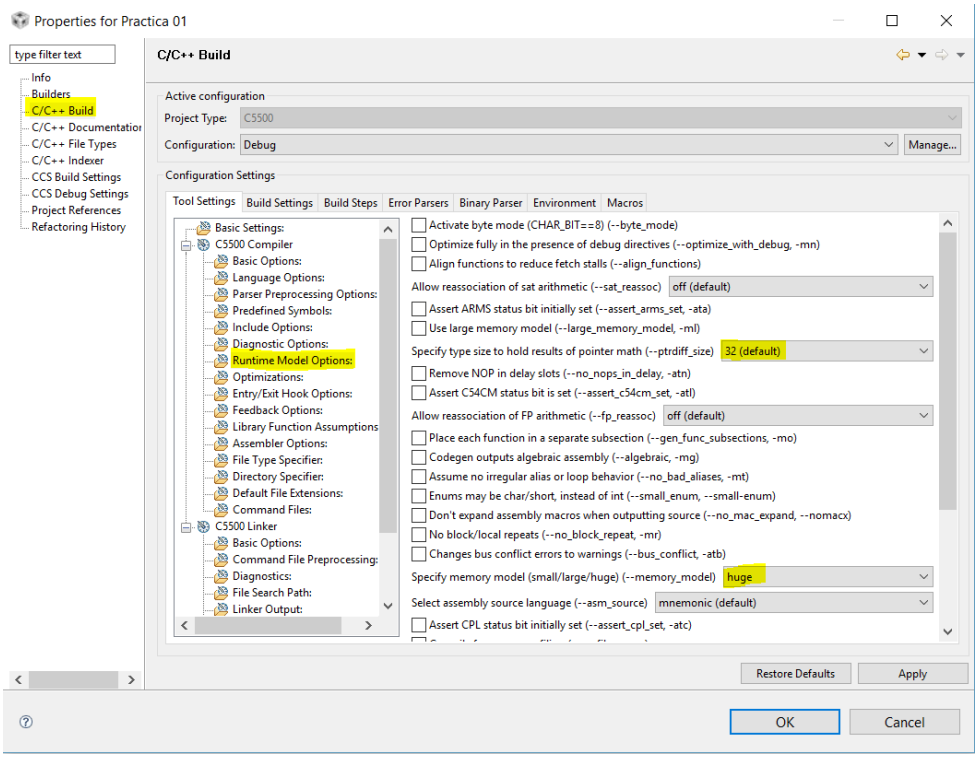


Ilustración 29: CCS configuración programa

Anexos

4. Una vez completado, compilaremos el proyecto:

Project->Build Active Project

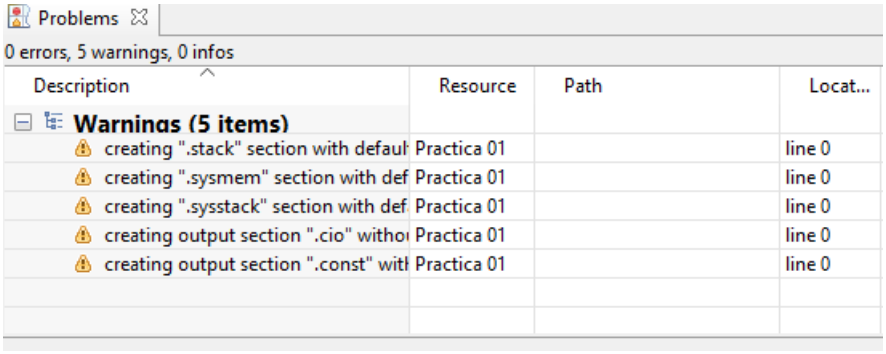


Ilustración 30: Problemas CCS

Donde cómo podemos ver, no tendremos ningún error, pero aparecerán distintos warnings que nos indican que necesitamos añadir al proyecto el mapa de memoria de nuestra tarjeta.

Por lo que copiaremos el archivo incluido en la carpeta del ejercicio, **c5505.cmd**, y lo copiaremos en la carpeta de proyecto, y volvemos a compilar.

Esta vez obtendremos 0 errores, y 0 warnings.

5. Ahora configuramos el enlace CCS-tarjeta: para ello:

New->Target Configuration File

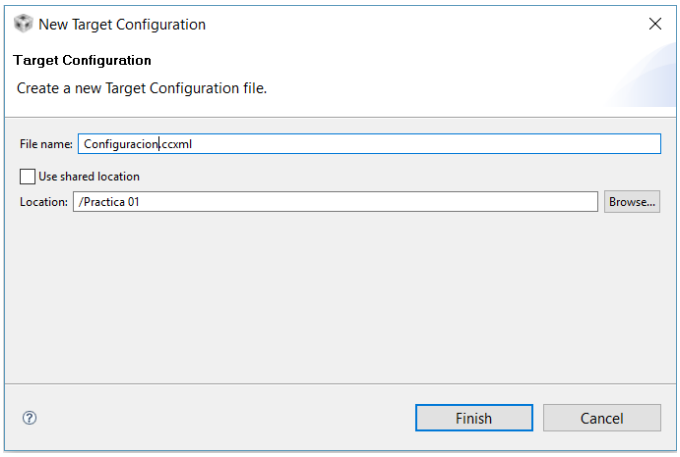


Ilustración 31: Target configuration

Aquí elegiremos el nombre del archivo, que guardaremos en el directorio del proyecto con la extensión **.ccxml**.

Anexos

Configuraremos nuestro dispositivo como viene a continuación:

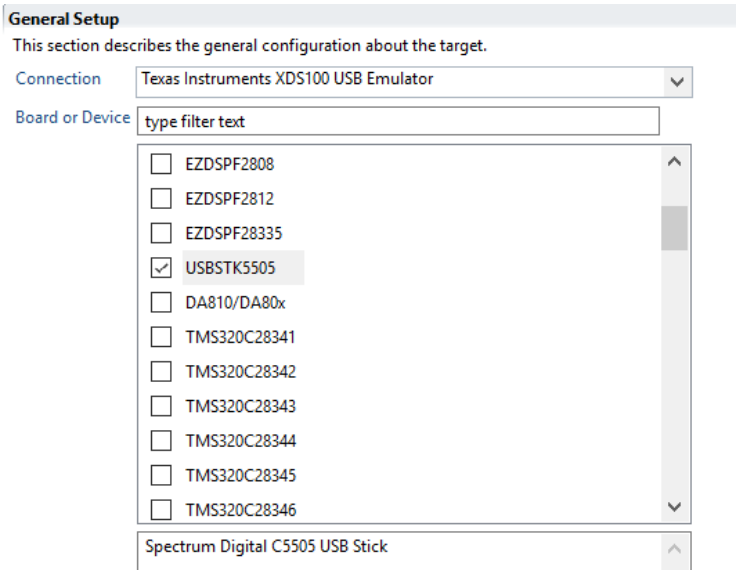


Ilustración 32: selección de DSP

6. Tras esto realizaremos la **carga del programa en la tarjeta**, pulsaremos sobre el icono de Debug  y el programa se conectara con nuestra tarjeta usando este archivo, compilará los archivos y los cargará en la tarjeta.

También nos abrirá el entorno de depuración donde podremos ver el funcionamiento del programa.

Donde para ejecutar, bastará con darle al **"play"** si lo queremos ejecutarlo entero, o paso a paso mediante las flechas.

Para terminar la ejecución tendremos un botón de **Stop**.

Anexos

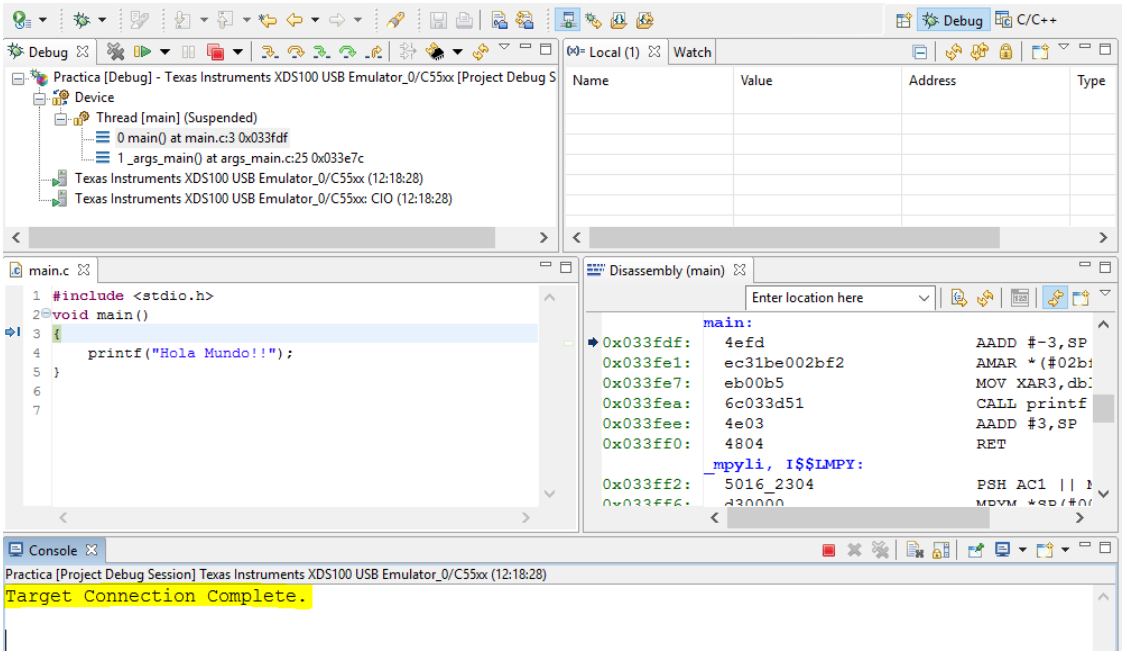


Ilustración 33: CCS ejecución

En el caso de querer volver a ejecutar el programa utilizaremos el icono de “reload program”.

Tras ejecutar nuestro programa veremos en la consola el resultado de nuestro programa:

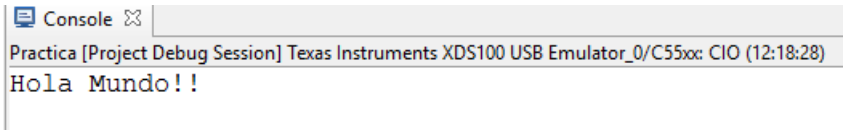


Ilustración 34: CCS resultado

Recibiremos el “Hola Mundo!!” introducido en la sección de código, con lo que habremos terminado nuestra primera toma de contacto con CCS.

Anexos

Matlab:

Introducción:

Para realizar un primer contacto con Matlab o un repaso general para los que ya lo han utilizado con anterioridad. En esta práctica recorreremos las diferentes secciones de Matlab, y veremos cómo funciona.

Cuando abrimos la aplicación por primera vez, sin cargar ningún archivo ya creado, encontramos esta ventana, compuesta de 4 secciones, la primera se trata del directorio de trabajo, o workspace donde guardaremos nuestros archivos, es conveniente revisarlo cada vez que abrimos Matlab y guardar los archivos donde realmente queremos.

La segunda sección es "Comand Window" la ventana de comandos, donde podremos trabajar de manera directa para realizar cálculos o visualizaciones concretas, pero no se trata de una sesión de trabajo, sino un lugar donde poder ejecutar acciones en el momento.

La sección número tres nos muestra en el directorio de trabajo, los archivos contenidos, y si estamos trabajando en un fichero, lo guardará en dicho directorio.

Por último la cuarta sección, o el contenido del "workspace", aquí se almacenan las variables ejecutadas en el programa durante la sesión, y permanecerán activas mientras dure la sesión.

Además no se guardarán automáticamente con la sección de trabajo, por lo que deberemos volver a ejecutar nuestra sesión para volver a cargar las variables.

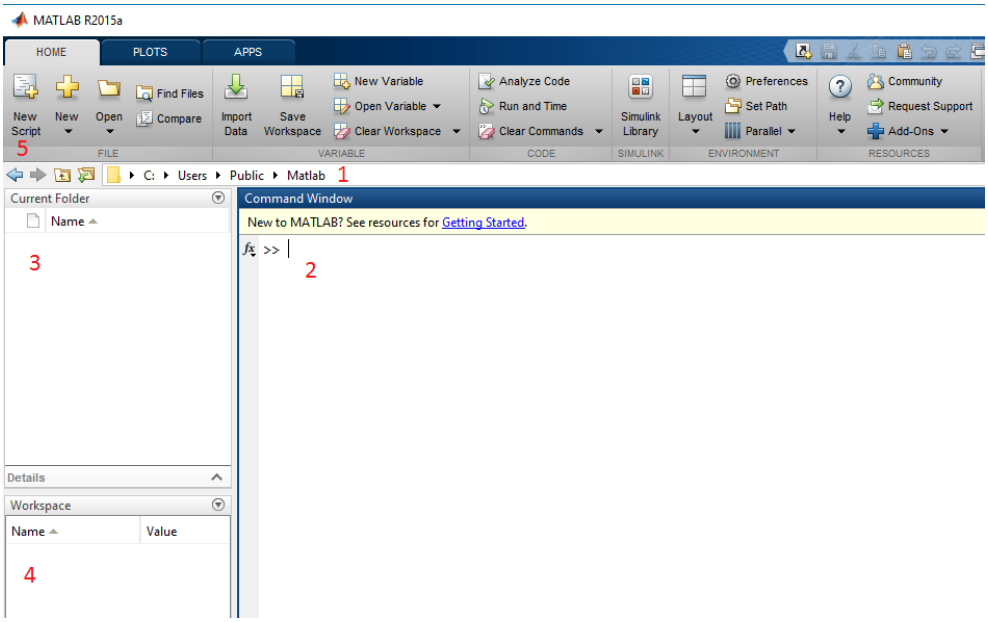


Ilustración 35: Apartados Matlab

Anexos

En último lugar he añadido un quinto punto, aunque no se trate de una sección. Cuando pulsamos en "New Script". Se nos abrirá una nueva sesión de trabajo en la parte superior de la "Comand Window", donde podremos almacenar código de tal manera que podamos guardarlo y ejecutarlo cuando deseemos lo que nos permitirá retomar nuestro trabajo.

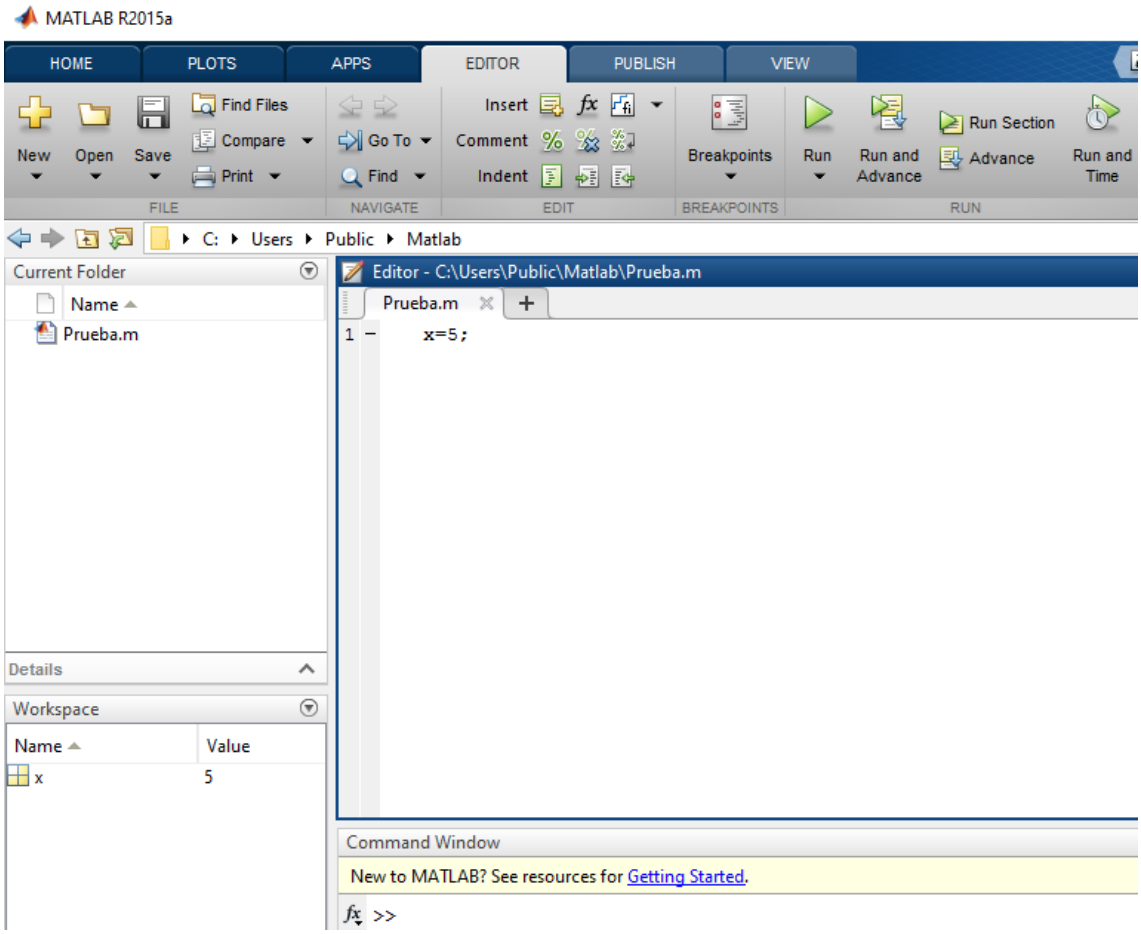
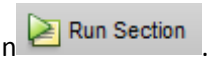


Ilustración 36: Prueba Matlab

Pasos a seguir para la realización de la práctica:

1. Pulsamos en el botón de Matlab de nuestro entorno Java. Se nos abrirá un script con código escrito organizado en distintas secciones.
2. Iremos leyendo los comentarios adquiriendo el modo de trabajo en secciones de código, cabe puntualizar que si modificamos el valor de las variables siempre se puede refrescar volviendo a ejercitar esa sección de código seleccionando la sección y pulsando en 

.

Anexos

3. En la segunda sección encontramos la forma de generar una senoidal y visualizarla en un gráfico.
4. En el tercer apartado modificaremos la senoidal para componerla con un ruido blanco, y también la visualizaremos en un gráfico para apreciar las diferencias.
5. Ahora realizaremos el mismo trabajo con una cosenoidal, la generaremos con una frecuencia menor, para ver como mejora la calidad de la señal respecto a la frecuencia de muestreo.
6. Y por último le añadiremos ruido al igual que con la senoidal.

Con esto habremos repasado cómo funciona el uso del almacenamiento de variables en el workspace, como ejecutar secciones de código, y como muestrear señales, que será parte de lo que utilizaremos en los siguientes experimentos.

Anexos

Filtrado Digital de Señal Con DSP

Ejercicios Bloque 1

Iñigo Rodríguez Martínez

Tutor: Antonio Moisés Zorzano Martínez

31/01/2017

Procesador de señal a emplear: **eZdsp TMS320VC5505 USB Stick**

UNIVERSIDAD DE LA RIOJA

A lo largo de este documento se plantean ejercicios a desarrollar por el alumno para continuar con el aprendizaje del procesado digital, no limitándonos solo a los experimentos realizados y actividades guidas. Ya que se asentarán mejor los conocimientos y se adquirirán nuevos poniéndolo en práctica.

Anexos

Introducción:

Se plantean en este documento una serie de ejercicios más o menos abiertos en función de los casos, que buscan un mayor manejo de la herramienta y la continuación del aprendizaje tras la parte guiada de los experimentos.

Se recomienda la lectura total de cada ejercicio antes de empezar a realizarlo ya que muchos de los objetivos pueden realizarse de manera simultánea.

Generación de señales y uso de MATLAB:

1. Genera 5 señales distintas, y grábalas como se ha explicado en la guía. Visualiza durante su generación, la señal en el osciloscopio, y su espectro frecuencial, añadiendo dos impresiones de pantalla, y una breve explicación de él porque tiene esa forma el análisis frecuencial.

2. Realiza un script de Matlab con el que poder analizar el espectro frecuencial de las señales generadas. Analiza las 5 señales generadas en el ejercicio anterior.
   Pista Comandos: La lectura de las señales se realiza con “Audioread”, pasamos de muestras a frecuencia con el comando “fft”. Con el comando “linspace” crearemos vectores necesarios para realizar la representación.

3. Analiza y modifica a tu gusto el script de Matlab correspondiente al ejercicio 3, que se encuentra: **C:\Archivos_TFG\Señales SoundCard Scope\Matlab**
   Para ello entenderemos como hemos generado las señales y como hemos generado el fichero .wav.
   Tras esto generaremos 5 ficheros distintos, compuestas cada una por más de una señal a diferentes frecuencias. Si la generamos con frecuencias menores a 8KHz, podremos usarlas en los posteriores experimentos.

4. Realizaremos el análisis frecuencial del ejercicio dos a las señales generadas, incluyendo capturas de pantalla de los resultados y haciendo un breve análisis escrito de su respuesta.

Anexos

Familiarización con CCS:

A continuación cambiaremos de SoundCard y Matlab para diseñar un pequeño proyecto de CCS, repasando así como es la creación de un proyecto y utilizando para el desarrollo del ejercicio nuestra tarjeta eZdsp.

5. El siguiente ejercicio nos auxiliaremos de la guía donde se desarrolla el proyecto" Hola Mundo" para no olvidar los pasos a realizar. Y los realizaremos de la misma manera hasta la hora de introducir código. Donde introduciremos todo menos el contenido del void main() {…}.

6. Se pide realizar un programa que lea caracteres introducidos desde el teclado, realizando tres funciones distintas en función del carácter introducido, siendo estas, una salida por pantalla, un bucle donde se introducirá por teclado un número por teclado y será esta el número de veces que se realice, y por ultimo un fallo en el carácter introducido.

Anexos

Anexo 2:

Filtrado Digital de Señal Con DSP

Filtrado FIR

Iñigo Rodríguez Martínez

Tutor: Antonio Moisés Zorzano Martínez

31/01/2017

Procesador de señal a emplear: **eZdsp TMS320VC5505 USB Stick**

UNIVERSIDAD DE LA RIOJA

A lo largo de este documento se describen los contenidos necesarios de procesado digital que necesitamos para abordar el filtrado FIR, y posibilitar el entendimiento de los experimentos que se realizaran a continuación.

Anexos

Contenido

Anexos

Introducción:

Este documento contendrá información relativa a los filtros FIR a nivel teórico, que nos ayudará a aprender o repasar las nociones principales relativas al uso de filtros FIR y su diseño.

Tener en cuenta que el diseño de filtros durante los experimentos será guiado, y que no habrá mayor dificultad en cuanto al uso de fórmulas, ya que nos asistiremos de MATLAB para el desarrollo de los filtros y la adquisición de sus coeficientes.

Pero podremos comprobar con los cálculos teóricos que los resultados son idénticos a los obtenidos y que en los ejercicios a realizar tras el experimento, se requerirán algunos de los conocimientos aquí presentados.

Filtrado FIR:

Es un acrónimo en inglés: "*Finite Impulse Response*" que significa: "*Respuesta finita al impulso*". Es un tipo de filtros digitales que presentan ante una señal impulso de entrada un número finito como resultado de términos no nulos.

Principales Características:

- Los filtros FIR son siempre estables y son capaces de tener una respuesta de fase lineal, por lo que su respuesta tiene un retraso constante.
- El diseño de filtros FIR requieren la selección de la secuencia que mejor representa la respuesta a impulso de un filtro ideal.
- Los filtros FIR no recursivos son menos sensibles a los efectos de longitud de palabra finita que los IIR, ofreciendo diversas ventajas en los cálculos que conlleva el filtrado.
- El mayor problema de los filtros FIR es que para unas especificaciones dadas requieren un filtro de orden mucho mayor que los filtros IIR.
- Si se considera al sistema causal, la expresión que caracteriza a un filtro FIR de coeficientes $\mathbf{b_k}$ es:

$$h[n]=\sum_{k=0}^{L-1} b_k \delta[n-k]$$

Anexos

- La salida se puede calcular mediante convolución directa de la entrada con la respuesta al impulso:

$$y[n] = \sum_{k=0}^{L-1} h[k]x[n-k]$$

- La función de sistema que caracteriza al filtro es la Transformada Z de la respuesta al impulso:

$$H(z) = \sum_{n=-\infty}^{\infty} h[n]z^{-n} = \sum_{k=0}^{L-1} b_k z^{-k}$$

- Su función de transferencia tiene denominador constante y solo tiene **ceros.**
- Los filtros FIR. son muy fáciles de realizar. La mayoría de los procesadores de señales digitales tienen unas arquitecturas internas que hacen factible su construcción.

La Fase Lineal:

Como se ha comentado en las propiedades una de las características más importantes que presentan los filtros FIR es la posibilidad de disponer de la fase en la respuesta en frecuencia perfectamente lineal. Aplicando un filtro a una señal se modifican sus características de amplitud y de fase. Esto dependerá del módulo y la fase de la respuesta en frecuencia que presente nuestro filtro. El retardo de fase y de grupo nos otorgan una medida de cómo el filtro va a llevar a cabo estas modificaciones.

Un filtro con una característica de fase no lineal provocará una distorsión de fase en la señal de entrada que lo atraviese, pues cada componente en frecuencia sufrirá un retardo no proporcional a dicha frecuencia, modificando así la relación entre armónicos. Se puede evitar empleando filtros con características de fase exactamente lineal en determinadas bandas de frecuencia.

Un filtro tiene fase de la respuesta en frecuencia lineal si ésta satisface cualquiera de las dos primeras relaciones, con α y β constantes.

Primera y segunda Condición:

$$\varphi(\Omega) = -\alpha\Omega$$
$$\varphi(\Omega) = \beta - \alpha\Omega$$

Anexos

Si el filtro tiene la respuesta al impulso con simetría par con respecto al punto central, satisface la primera condición y tendrá constantes el retardo de fase y de grupo.

$$h[n]=h[L-n-1] \qquad \begin{cases} n=0,1\ldots(L-1)/2 & \text{para L impar} \\ n=0,1\ldots(L/2-1) & \text{para L par} \end{cases}$$

En este caso la fase de la respuesta en frecuencia es función de la longitud del filtro que vemos en la cuarta relación.

$$\varphi(\Omega)=-\left(\frac{L-1}{2}\right)\Omega$$

Cuando satisfacen la segunda condición:

La respuesta al impulso presenta simetría impar, y el filtro sólo tiene retardo de grupo constante, con una fase de la respuesta en frecuencia.

$$h[n]=-h[L-n-1] \qquad \begin{cases} n=0,1\ldots(L-1)/2 & \text{para L impar} \\ n=0,1\ldots(L/2-1) & \text{para L par} \end{cases}$$

$$\varphi(\Omega)=\frac{\pi}{2}-\left(\frac{L-1}{2}\right)\Omega$$

La siguiente clasificación distingue los filtros FIR según su simetría y el número de coeficientes.

SIMETRÍA	número de coeficientes L	Tipo de filtro
PAR	IMPAR	I
h[n]=h[L-n-1]	PAR	II
IMPAR	IMPAR	III
h[n]=-h[L-n-1]	PAR	IV

Es importante indicar que en los cuatro tipos de filtros se puede demostrar que si aparece un cero en cualquier punto $\mathbf{z_0}$ del plano z, debe ir acompañado de otro recíproco conjugado

$(1/z_0)$. Si además la respuesta al impulso es real, la función del sistema H(z) será una función racional con coeficientes reales, y cada cero debe ir acompañado por su conjugado. En

consecuencia, en este tipo de sistemas los ceros aparecen por cuartetos recíprocos conjugados con las siguientes excepciones:

1. Los ceros sobre la circunferencia unidad (salvo en z=±1) aparecen por parejas, ya que coinciden con sus recíprocos.
2. Los ceros reales no situados en la circunferencia unidad aparecen por parejas, pues coinciden con sus conjugados.
3. Los ceros en z=±1 coinciden con su recíproco y su conjugado, pudiendo aparecer solos.

Se puede demostrar que en los filtros tipo II debe aparecer un cero en z=-1, lo que significa que no pueden realizar características paso alto, que en los filtros tipo III deben aparecer ceros en Z=1 y z=-1, con lo que no sirven para sistemas que permitan el paso de las bajas y/o de las altas frecuencias, y que para los tipo IV el cero obligatorio se sitúa en z=1, con lo que no se pueden emplear como filtros paso bajo.

Métodos de diseño de filtros FIR:

Existes tres métodos de diseño de filtros FIR:

- Método de las Series de Fourier.
- Método del Muestreo en Frecuencia.
- Métodos Iterativos basados en condiciones óptimas de diseño

En nuestro experimento, nos encontraremos en el tercer método. Este alberga los distintos algoritmos que se han desarrollado a lo largo de la historia de métodos específicos para diseñar filtros.

En nuestro caso nos hemos decantado por el algoritmo de Parks-McClellan (algoritmo de intercambio Remez) que tiene una implementación muy sencilla en MATLAB y será idóneo para una iniciación al diseño de filtros.

```
% firpm(x,y,z)
% x -> numero de coeficientes del filtro mas 1, en nuestro caso lo fijamos
%      a 47 para tener 48.
% y -> tanto por uno, desde 0 a la frecuencia de muestreo partido por 2,
%      selecciona las frecuencias reseñables
% z -> debe tener el mismo numero de coeficientes que "y", y define la
%      magnitud de la respuesta en las frecuencias seleccionadas.
```

Anexos

Tipos de Filtros:

Encontramos cuatro tipos de filtros principales **tanto para FIR como para IIR** en cuanto al tratamiento de la señal de entrada:

-Los filtros pasa-bajos, pasa-altos, pasa-banda y los para-banda.

Filtro pasa-bajos:

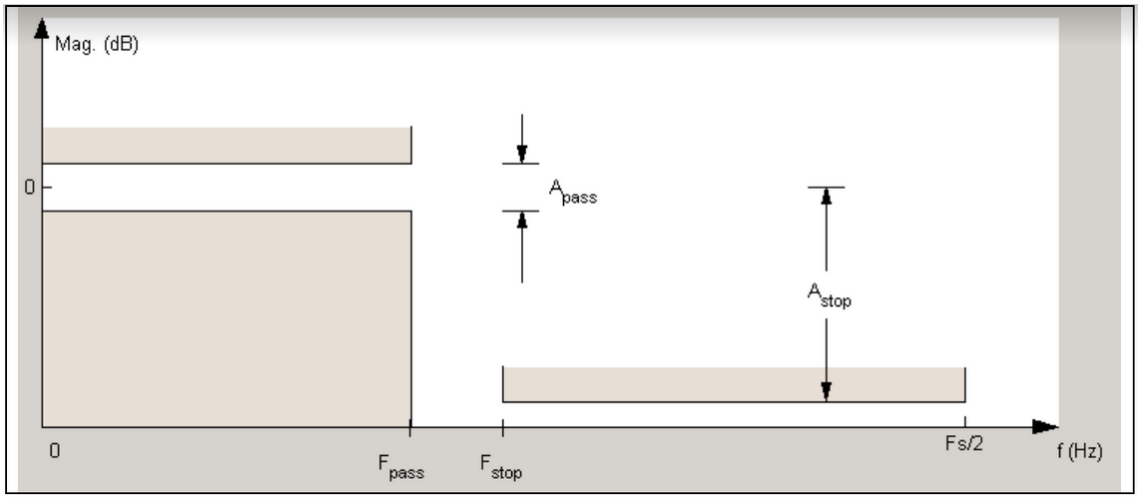


Ilustración 37: Filtro pasa-bajos

Donde *Apass* es el rizado de paso tolerable, *Astop* la atenuación en la banda de rechazo. Y en la banda de transición formada por la frecuencia de paso *Fpass* y la frecuencia de rechazo *Fstop*, denominada banda de transición.

Filtro pasa-altos:

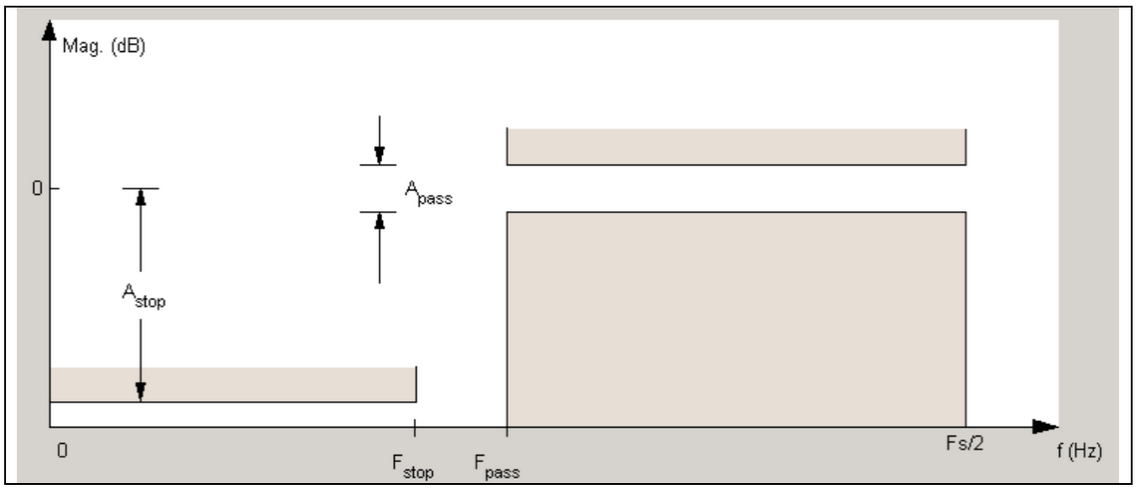


Ilustración 38: Filtro pasa-altos

Aquí vemos a Fs/2 como la máxima frecuencia de paso debido al muestreo de la señal.

Anexos

Filtro pasa-banda:

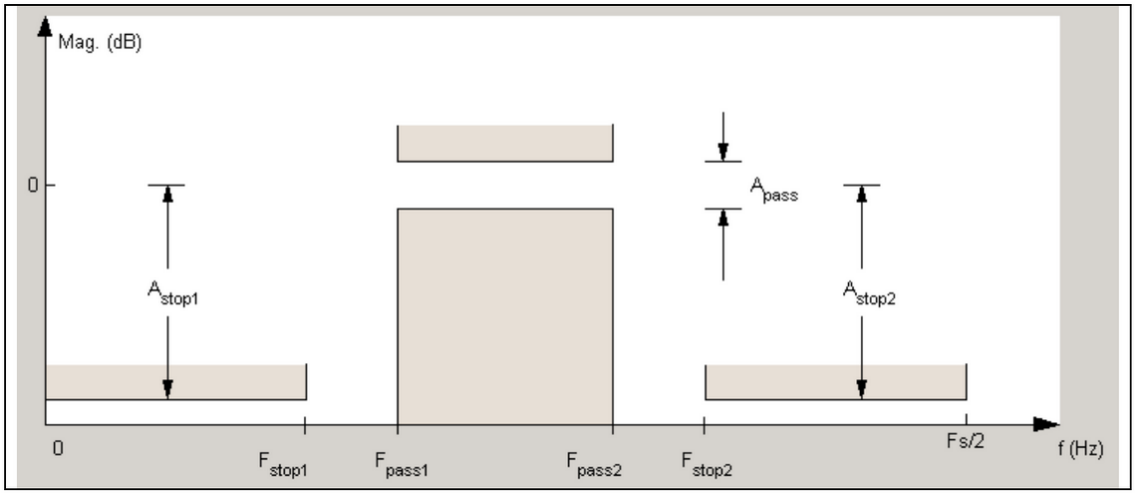


Ilustración 39: Filtro pasa-banda

Filtro para-banda:

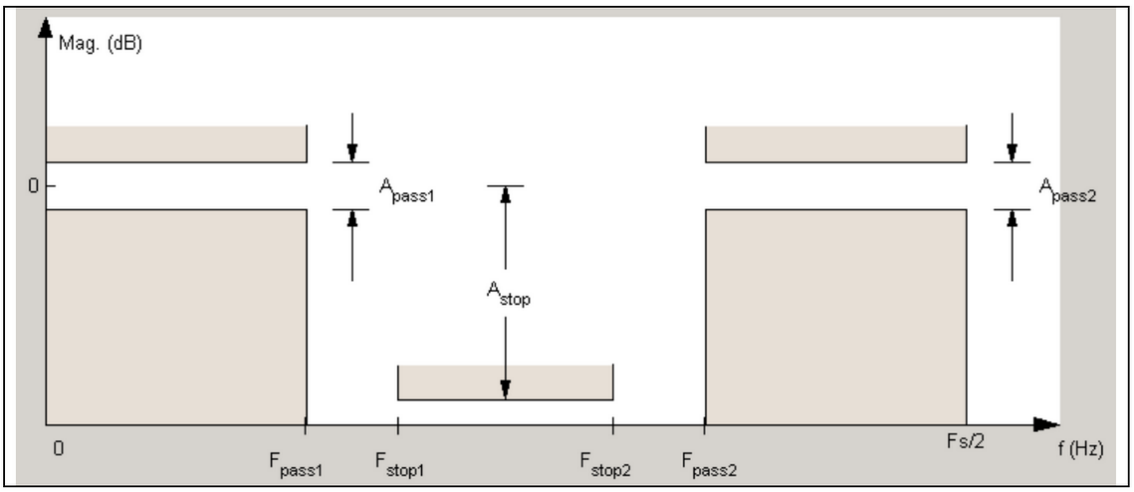


Ilustración 40: Filtro para-banda

Anexos

Filtrado Digital de Señal Con DSP

Guía FIR

Iñigo Rodríguez Martínez

Tutor: Antonio Moisés Zorzano Martínez

31/01/2017

Procesador de señal a emplear: **eZdsp TMS320VC5505 USB Stick**

UNIVERSIDAD DE LA RIOJA

A lo largo de este documento se describen los pasos a seguir necesarios para la realización del experimento de procesado digital que desarrollara un filtrado FIR de principio a fin. Siendo este experimento real y realizado en nuestro DSP seleccionado.

Anexos

Contenido

Anexos

Introducción:

Tras la lectura del documento de Información General FIR, en esta guía encontraremos los pasos a seguir para la utilización de todos los programas que se utilizarán en el desarrollo del filtrado FIR planteado como experimento.

El alumno en un primer momento deberá simplemente seguir los pasos descritos y limitarse a adquirir los conceptos y la metodología que se describe a continuación.

Si no se siguen los pasos se podrán dar errores de compilación u otros resultados finales que nos son los que busca presentar esta guía, por lo que se realizarán las pruebas una vez terminado el seguimiento de los pasos.

Recordamos que en la columna de filtrado FIR se recorrerá de arriba abajo, ya que los pasos siguen un orden lógico para conseguir el resultado deseado.

Selección del tipo de filtro:

Como hemos visto en el último apartado de información general de filtrado FIR, encontramos diferentes tipos de filtros en función del tipo de tratamiento que queramos dar a la señal.

Nuestra herramienta mostrará los 4 tipos de filtrado en el siguiente recuadro tras la guía, estos contendrán diferentes scripts de **MATLAB** realizados con el algoritmo de Parks McCellan.

Pasos Primer bloque:

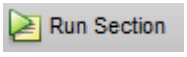
1. Elegiremos el filtrado Pasa-banda ya que es el más representativo a nivel visual de la respuesta. Cuando clicamos se nos abrirá MATLAB, con un script ya escrito.

2. Leeremos detenidamente el script tratando de comprender su funcionamiento con ayuda de los comentarios incluidos, y en el caso de encontrar dudas no resultas en los comentarios, podremos utilizar la ayuda de Matlab donde simplemente escribiendo en la "Comand Window" help y el nombre de la función que no entendemos podremos acceder a la ayuda de Matlab.
   Habrá que calcular las frecuencias reseñables sabiendo que la frecuencia de muestreo a la que se realizará el experimento es **8000 Hz.**

Anexos

```
Command Window
>> help firpm
firpm - Parks-McClellan optimal FIR filter design

    This MATLAB function returns row vector b containing the n+1 coefficients of the
    order n FIR filter whose frequency-amplitude characteristics match those given
    by vectors f and a.

    b = firpm(n,f,a)
    b = firpm(n,f,a,w)
    b = firpm(n,f,a,'ftype')
    b = firpm(n,f,a,w,'ftype')
    b = firpm(...,{lgrid})
    [b,err] = firpm(...)
    [b,err,res] = firpm(...)
    b = firpm(n,f,@fresp,w)
    b = firpm(n,f,@fresp,w,'ftype')

    Reference page for firpm

    See also butter, cfirpm, cheby1, cheby2, ellip, fir1, fir2, fircls, fircls1,
        firls, firpmord, function_handle, rcosdesign, yulewalk
```

Ilustración 41: ayuda firpm

3. Tras entender los comandos incluidos en el script, procederemos a ejecutar el script, dado que solo tiene una sección bastará con pulsar en: 

para ejecutarlo todo.

4. Cuando termine de ejecutarlo, veremos que nos abre una ventana **FVTOOL** un entorno de visualización donde veremos la forma de nuestro filtro. Siendo el eje X la frecuencia normalizada en formato (rad/muestra) y el eje y la atenuación en db.
   Analizando el resultado vemos claramente que no se trata de un filtro ideal, sino que posee un rizado y unas bandas de transición donde el resultado dependerá de cómo de cerca o de lejos nos encontremos de las frecuencias reseñables ya que estas sí que están bastante bien fijadas y con un valor en magnitud adecuado.

5. Volviendo a Matlab veremos cómo se han generado y guardado variables en nuestro Workspace. Donde **b**, será el resultado del filtrado, los 48 coeficientes a aplicar a una señal para realizar el filtro visualizado en la FVTool. Y **b_int** será el mismo resultado pero normalizado al formato **16 bits entero** que empleará nuestro DSP para realizar sus operaciones.

6. Pulsar en las variables en el workspace para ver sus valores y verificar su validez. También podemos escribir el nombre de la variable en la ventana de comandos para que nos muestre sus resultados.

7. Dejaremos abierto Matlab o copiamos los coeficientes de la variable “b_int” a un block de notas para no perder los resultados.

Anexos

8. En primer lugar, pulsando sobre el primer botón “Introducir coeficientes”, se nos abrirá por defecto o seleccionaremos un editor de texto como puede ser block de notas, para ver los coeficientes que están introducidos en el programa por defecto.

9. Comprobaremos si son los mismos, y en el caso contrario en futuros ejercicios donde realicemos otro tipo de filtros, los sustituiremos por los adquiridos en Matlab. Guardaremos el fichero y cerramos el editor de textos y Matlab.

Cuando sustituimos los coeficientes del filtro por otros, no necesitaremos introducir todo lo que viene en el fichero por defecto que se realizó de la forma más pulcra:

Int16 firCoefFixedPoint[NUM_TAPS]={ (Int16)(-.0011606*32767.0),(Int16)(0.005235*32767.0), (Int16)(0.0019751*32767.0), (Int16)(-0.0010696*32767.0)… };

En nuestro caso será válido con introducir los coeficientes de una forma más sencilla para ahorrar tiempo, copiaremos y pegaremos todos, los separaremos por comas, y los encerraremos a todos ellos entre dos corchetes. Con esto será suficiente.

Int16 firCoefFixedPoint[NUM_TAPS]= {99, -55, -181,-85, 127, 179, 56, -141, -311,… };

Filtrado con DSP:

Llegados a este punto accedemos al siguiente cuadro, avanzando así con el experimento. Ya tenemos diseñado nuestro filtro y hemos introducido los coeficientes en el programa utilizado en el experimento FIR.

Pasos segundo bloque:

1. Tendremos en primer lugar la opción de cambiar la señal de entrada de audio a filtrar. Para ello pulsaremos en el siguiente botón “Selección de entrada” donde se nos abrirá un carpeta que contiene señales de entrada de audio generadas por la SoundCard Scope, que podremos introducir. Sin cerrar este pulsaremos en el botón introducir entrada y se nos abrirá la carpeta donde se encuentra introducida la señal a procesar en el experimento en formato .wav.

2. Cogeremos la señal de la carpeta de entradas llamada “Default” y la copiaremos en la donde introduciremos la entrada, sustituyendo su nombre por **input.wav** y borrando la que había en la carpeta de destino con el mismo nombre.

3. En este punto ya tenemos todas las entradas listas, por lo que procedemos a conectar el Hardware necesario: **eZdsp TMS320VC5505 USB Stick** en una de las entradas usb disponibles en el pc.

Anexos

4. Pulsamos en el botón “Ejecutar EzDSP” aquí se nos abrirá CCS (Code Composer Studio), que nos hará seleccionar un directorio de trabajo, donde seleccionaremos en la carpeta de los archivos de la herramienta la llamada “Experimento FIR” como Workspace.

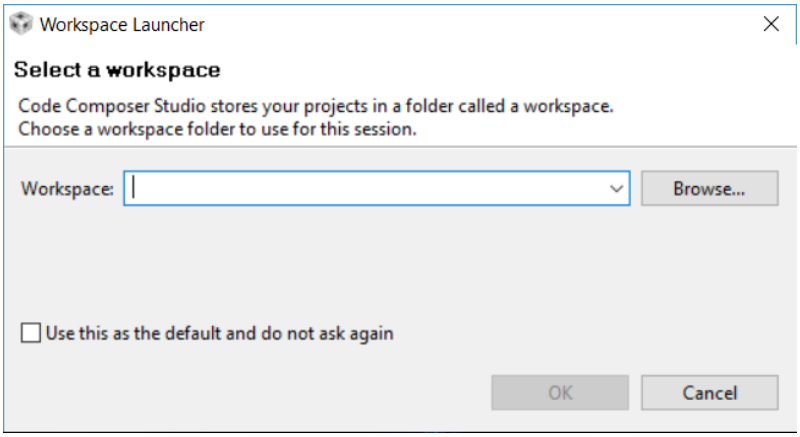
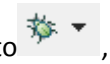


Ilustración 42: Selección de workspace

5. El experimento está ya preparado, por lo que intentaremos comprender lo que se realiza en él, en los tres archivos que nos ha abierto:
   El primero corresponde al procesado de la señal **fixedPointFir.c**, el segundo es el que nos enlazará las distintas secciones pidiendo la introducción de comandos por pantalla para la realización del experimento **fixedPointFirTest.c**, y el tercero son los coeficientes calculados introducidos en el paso 2 de este bloque **firCoef.h**.

6. Ahora compilaremos y ejecutamos el proyecto  , compilará y si no hay ningún problema de compilación ni warnings ni errores nos abrirá la ventana de depuración.

7. Donde para ejecutar, bastará con darle al **“play”**, tras esto atenderemos a la ventana de comandos donde nos pedirá que seleccionemos el tipo de señal de entrada a procesar. Tras esto esperaremos hasta que nos devuelva que el experimento ha sido completado.
   Para terminar la ejecución bastará con pulsar el botón de **Stop.**

8. Cerraremos CCS y clicamos en la siguiente opción de nuestra herramienta “Comprobación de salida” que nos abrirá la carpeta con la señal de salida procesada y podremos escucharla para comprobar su tratamiento.

Anexos

Comprobación de salida:

Tras la comparación de salida auditiva donde, notaremos diferencia con la señal de entrada dado que en el experimento realizado eliminaremos dos de las senoidales de las tres que se componía la señal de entrada inicial.

Pasaremos ahora de realizar una comprobación más exhaustiva, donde podremos ver como se ha filtrado y por tanto nos dará una mayor cantidad de datos y un mejor tipo de análisis ya que de oído no podemos diferenciar las frecuencias filtradas o las que no.

Pulsaremos en el siguiente apartado "Comprobación MATLAB", este nos devolverá al programa **MATLAB**, donde nos abrirá otro script ya implementado para la lectura de archivos **.wav**, y la visualización de los resultados del filtrado.

Pasos tercer bloque:

1. Para que el script funcione correctamente, hemos de colocar el workspace de Matlab en la carpeta donde se encuentra el script, que se encuentra en la misma carpeta donde depositamos las señales de audio en la carpeta de experimento de CCS.
   …/Archivos_TFG\Experimentos\Experimento_FIR\EXP_fixed_point_fir\data
2. A continuación realizaremos una lectura del script intentando comprender los comandos introducidos y buscando en la ayuda lo que no entendemos.
3. Ejecutaremos el script sección a sección analizando los resultados en el apartado de variables de MATLAB, comprendiendo los valores obtenidos en función de las señales introducidas.
4. Por ultimo al ejecutar la última sección del script, se nos abrirá un gráfico con la señal sin filtrar "original" y la señal tratada "filtrada" donde podremos comprobar el trabajo del filtro y su correcto funcionamiento.

Anexos

Aprendizaje Autónomo:

Una vez terminada la realización y cogido la metodología empleada en el primer experimento, volveremos al principio del bloque de "Filtrado FIR" y volveremos a recorrer los distintos pasos de la herramienta, seleccionando otro tipo de filtro, modificando las frecuencias reseñables, para modificar donde se realizaran los cambios en el filtrado.

También tendremos la oportunidad de cambiar la señal de entrada, con otras frecuencias, tomando las generadas con la **SoundCard Scope**, en el ejercicio del primer bloque o generar una señal en el momento para que corresponda con las frecuencias de filtrado introducidas en el filtro.

Cuando ejecutemos todos los pasos del filtrado obtendremos los resultados acordes al filtrado realizado y la señal introducida obteniendo una señal nueva fruto del procesado real de la señal en el DSP en **Code Compose Studio**.

A la cual podremos analizar con el script de **MATLAB** de la misma manera con la que se comprobó la señal de salida del experimento guiado.

Nótese que si realizamos por ejemplo un filtro pasa-bajos a una frecuencia introducida por el alumno y la señal de entrada a filtrar solo se compone de una frecuencia en la que el filtro deja pasar la señal, el filtro como es evidente no realizará ninguna operación. Y por lo tanto obtendremos una señal de salida idéntica a la de entrada.

Anexos

Filtrado Digital de Señal Con DSP

Ejercicios FIR

Iñigo Rodríguez Martínez

Tutor: Antonio Moisés Zorzano Martínez

31/01/2017

Procesador de señal a emplear: **eZdsp TMS320VC5505 USB Stick**

UNIVERSIDAD DE LA RIOJA

A lo largo de este documento se plantean ejercicios a desarrollar por el alumno para continuar con el aprendizaje del procesado digital, no limitándonos solo a los experimentos realizados y actividades guidas. Ya que se asentarán mejor los conocimientos y se adquirirán nuevos poniéndolo en práctica.

Anexos

Introducción:

Se plantean en este documento una serie de ejercicios más o menos abiertos en función de los casos, que buscan un mayor manejo de la herramienta y la continuación del aprendizaje tras la parte guiada de los experimentos.

Se recomienda la lectura total de cada ejercicio antes de empezar a realizarlo ya que muchos de los objetivos pueden realizarse de manera simultánea.

Ejercicios planteados:

1. Partiendo de los script de Matlab incluidos en el programa, mediante el método de Parks McCellan, se calcularán, teniendo en cuanta que la frecuencia de muestreo es **8 KHz**, los siguientes filtros con las siguientes características:
    a) Filtro pasa-bajos: con frecuencias reseñables (200, 500, 1000, 1200, 2000, 3000) donde la frecuencia de paso será 1000 Hz.
    b) Filtro pasa-altos: con frecuencias reseñables (100,500, 1200, 2000, 2500, 3500) donde la frecuencia de paso será 1200 Hz.
    c) Filtro pasa-banda; con frecuencias reseñables (400, 800, 1300, 2200, 2900, 3200, 3700) donde la frecuencia de paso serán 1300 y 2900 Hz.
    d) Filtro para banda: con frecuencias reseñables (800, 1300, 1600, 2200, 3000) donde la frecuencias de paro 1300 y 2200 Hz.

2. Para el siguiente ejercicio necesitaremos recordar cuales eran los métodos de diseño de filtros FIR. Tras esto el alumno deberá investigar de forma autónoma por internet y apoyándose en la ayuda de Matlab para conocer y aprender cómo utilizar otras funciones para el diseño de filtros FIR en los distintos métodos.
   Se pide conocer los comandos y el uso:
    a) Métodos de creación de filtros mediante ventanas:
       -Rectangular, Bartlett, Blackman, Hanning, Hamming y Kaiser.
       -fir1
    b) Métodos de creación de filtros mediante la frecuencia de muestreo:
       -fir2

3. Se realizará un script con tantas secciones de código como tipos de filtros anteriormente nombrados, con un ejemplo de cada uno de los tipos con comentarios sobre los parámetros que le debemos pasar a las distintas funciones.

Anexos

Filtrado Digital de Señal Con DSP

Código reseñable del experimento

Iñigo Rodríguez Martínez

Tutor: Antonio Moisés Zorzano Martínez

31/01/2017

Procesador de señal a emplear: **eZdsp TMS320VC5505 USB Stick**

UNIVERSIDAD DE LA RIOJA

En este documento se introducirá el código reseñable de los programas que merezca la pena destacar de tal manera que queden reflejados en el documento.

Anexos

Código CCS:

- Coeficientes FIR

```
Int16 firCoefFixedPoint[NUM_TAPS]={
(Int16)(-0.0011606*32767.0),(Int16)(0.005235*32767.0),  (Int16)(0.0019751*32767.0), (Int16)(-0.0010696*32767.0),
(Int16)(0.00070486*32767.0),(Int16)(-0.0023019*32767.0),(Int16)(-0.0085149*32767.0),(Int16)(0.0032251*32767.0),
(Int16)(0.019339*32767.0),  (Int16)(0.0019924*32767.0), (Int16)(-0.024454*32767.0), (Int16)(-0.00927*32767.0),
(Int16)(0.015833*32767.0),  (Int16)(0.0056414*32767.0), (Int16)(0.0040532*32767.0), (Int16)(0.022213*32767.0),
(Int16)(-0.018912*32767.0), (Int16)(-0.074497*32767.0), (Int16)(0.0079832*32767.0), (Int16)(0.13193*32767.0),
(Int16)(0.03699*32767.0),   (Int16)(-0.16479*32767.0),  (Int16)(-0.10113*32767.0),  (Int16)(0.15292*32767.0),
(Int16)(0.15292*32767.0),   (Int16)(-0.10113*32767.0),  (Int16)(-0.16479*32767.0),  (Int16)(0.03699*32767.0),
(Int16)(0.13193*32767.0),   (Int16)(0.0079832*32767.0), (Int16)(-0.074497*32767.0), (Int16)(-0.018912*32767.0),
(Int16)(0.022213*32767.0),  (Int16)(0.0040532*32767.0), (Int16)(0.0056414*32767.0), (Int16)(0.015833*32767.0),
(Int16)(-0.00927*32767.0),  (Int16)(-0.024454*32767.0), (Int16)(0.0019924*32767.0), (Int16)(0.019339*32767.0),
(Int16)(0.0032251*32767.0), (Int16)(-0.0085149*32767.0),(Int16)(-0.0023019*32767.0),(Int16)(0.00070486*32767.0),
(Int16)(-0.0010696*32767.0),(Int16)(0.0019751*32767.0), (Int16)(0.005235*32767.0),  (Int16)(-0.0011606*32767.0)
};
```

Ilustración 43: Coeficientes FIR

- Programa principal:

```
#include <stdlib.h>
#include <stdio.h>
#include "tistdtypes.h"
#include "fixedPointFir.h"

/* Define DSP system memory map */
#pragma DATA_SECTION(firCoefFixedPoint, ".data:fir");
#pragma DATA_SECTION(w, ".bss:fir");

#include "firCoef.h"

Int16 w[NUM_TAPS];
Int16 x[NUM_DATA],      // Input data
      y[NUM_DATA];      // Output data

void main()
{
    FILE  *fpIn,*fpOut;
    Int16 i,k,c,
          index;        // Delay line index
    Int8  temp[NUM_DATA*2];
    Uint8 waveHeader[44];

    printf("Exp --- Fixed-point_Block FIR filter experiment\n");
    printf("Enter 1 for using WAV file and START EXPERIMENT\n");
    scanf ("%d", &c);

    if (c == 1)
    {
        fpIn = fopen("..\\data\\input.wav", "rb");
        fpOut = fopen("..\\data\\output.wav", "wb");
    }
    else
    {
        printf("Cancelled, Restart Experiment");
        exit(0);
    }
```

Anexos

```
52      if (fpIn == NULL)
53      {
54          printf("Can't open input file\n");
55          exit(0);
56      }
57
58      if (c == 1)
59      {
60          fread(waveHeader, sizeof(Int8), 44, fpIn);
61          fwrite(waveHeader, sizeof(Int8), 44, fpOut);
62      }
63
64      // Initialize for filtering process
65      for (i=0; i<NUM_TAPS; i++)
66      {
67          w[i] = 0;
68      }
69      index = 0;
70
71      // Begin filtering the data
72      while (fread(temp, sizeof(Int8), NUM_DATA*2, fpIn) == (NUM_DATA*2))
73      {
74          for (k=0, i=0; i<NUM_DATA; i++)
75          {
76              x[i] = (temp[k]&0xFF)|(temp[k+1]<<8);
77              k += 2;
78          }
79          // Filter the data x and save output y
80          fixedPointBlockFir(x, NUM_DATA, firCoefFixedPoint, NUM_TAPS, y, w, &index);
81
82          for (k=0, i=0; i<NUM_DATA; i++)
83          {
84              temp[k++] = (y[i]&0xFF);
85              temp[k++] = (y[i]>>8)&0xFF;
86          }
87          fwrite(temp, sizeof(Int8), NUM_DATA*2, fpOut);
88      }
89
90      fclose(fpIn);
91      fclose(fpOut);
92
93      printf("\nExp --- completed\n");
94  }
95
```

Ilustración 44: Programa principal CCS FIR

Anexos

- Añadido para el filtrado

```
#include "tistdtypes.h"
#include "fixedPointFir.h"

/* Define DSP system memory map */
#pragma CODE_SECTION(fixedPointBlockFir, ".text:fir");

void fixedPointBlockFir(Int16 *x, Int16 blkSize,
                        Int16 *h, Int16 order,
                        Int16 *y,
                        Int16 *w, Int16 *index)
{
    Int16 i,j,k;
    Int32 sum;
    Int16 *c;

    k = *index;
    for (j=0; j<blkSize; j++)               // Block processing
    {
        w[k] = *x++;                        // Get the current data to delay line
        c = h;
        for (sum=0, i=0; i<order; i++)      // FIR filter processing
        {
            sum += *c++ * (Int32)w[k++];
            if (k == NUM_TAPS)              // Simulate circular buffer
            {
                k = 0;
            }
        }
        sum += 0x4000;                      // Rounding
        *y++ = (Int16)(sum>>15);            // Save filter output

        if (k-- <=0)                        // Update index for next time
        {
            k = NUM_TAPS-1;
        }
    }
    *index = k;                             // Update circular buffer index
}
```

Ilustración 45: Programa CCS filtrado FIR

Anexos

Matlab:

- Ejemplo de filtro

```
%% Filtrado FIR - Pasa bajos %%
% Filtro pasa bajos basado en el metodo PARKS-MACCELLAN
% firpm(x,y,z)
% x -> numero de coeficientes del filtro mas 1, en nuestro caso lo fijamos
%      a 47 para tener 48.
% y -> tanto por uno, desde 0 a la frecuencia de muestreo partido por 2,
%      selecciona las frecuencias reseñables
% z -> debe tener el mismo numero de coeficientes que "y", y define la
%      magnitud de la respuesta en las frecuencias seleccionadas.

f=[0 0.3 0.4 0.5 0.6 1];
m=[1 1 0 0 0 0];
% La frecuencia de muestreo a la que se realizara el experimento es: 8 KHz.
b=firpm(47 ,f , m);
b_int=round(32767*b); % Normalizado a 16 bits entero
fvtool(b,1);
```

Ilustración 46: Filtro ejemplo matlab

- Código de test

```
%% Lectura de las señales de audio in/out
[a, Fsr]=audioread('input.wav');
L=length(a);

[a1, Fsr1]=audioread('output.wav');
L1=length(a1);
%% Tratamiento para su representación
NFFT= 2^nextpow2(L);
Y=fft(a,NFFT)/L;
f=Fsr/2*linspace(0,1,NFFT/2+1);

NFFT1= 2^nextpow2(L1);
Y1=fft(a1,NFFT1)/L1;
f1=Fsr1/2*linspace(0,1,NFFT1/2+1);
%% Muestreo de las señales
subplot(2,1,1)
plot(f, 2*abs(Y(1:NFFT/2+1)))
legend ('Señal original')
subplot(2,1,2)
plot(f1, 2*abs(Y1(1:NFFT1/2+1)),'r') %señal en frecuencia
legend ('Señal filtrada')
```

Ilustración 47: Código de test matlab

Anexos

Anexo 3:

Filtrado Digital de Señal Con DSP

Filtrado IIR

Iñigo Rodríguez Martínez

Tutor: Antonio Moisés Zorzano Martínez

31/01/2017

Procesador de señal a emplear: **eZdsp TMS320VC5505 USB Stick**

UNIVERSIDAD DE LA RIOJA

A lo largo de este documento se describen los contenidos necesarios de procesado digital que necesitamos para abordar el filtrado IIR, y posibilitar el entendimiento de los experimentos que se realizaran a continuación.

Anexos

Contenido

Anexos

Introducción:

Este documento contendrá información relativa a los filtros IIR a nivel teórico, que nos ayudará a aprender o repasar las nociones principales relativas al uso de filtros IIR y su diseño.

Tener en cuenta que el diseño de filtros durante los experimentos será guiado y que no habrá mayor dificultad en cuanto al uso de fórmulas, ya que nos asistiremos de MATLAB para el desarrollo de los filtros y la adquisición de sus coeficientes.

Pero podremos comprobar con los cálculos teóricos que los resultados son idénticos a los obtenidos en Matlab y que en los ejercicios a realizar tras el experimento, se requerirán algunos de los conocimientos aquí presentados.

Filtrado IIR:

Es un acrónimo en inglés: "*Infinite Impulse Response*" que significa: "*Respuesta infinita al impulso*". En este tipo de filtros digitales la salida tendrá un número infinito de términos no nulos.

Son sistemas cuya salida depende además de salidas anteriores y que, estando en reposo, al ser estimulados con una entrada impulsional su salida no volverá al reposo, de ahí el calificativo de filtros de respuesta impulsional infinita (IIR). La ecuación en diferencias general es de la forma:

$$y(n) = b_0 x(n) + b_1 x(n-1) + \ldots + b_M x(n-M) - a_1 y(n-1) - a_2 y(n-2) - \ldots - a_N y(n-N) =$$
$$= \sum_{k=0}^{M} b_k \cdot x(n-k) - \sum_{k=1}^{N} a_k \cdot y(n-k)$$

Principales Características:

- Los filtros IIR producen en general distorsión de fase, por lo que la fase no es lineal con la frecuencia. Existen métodos de diseño para intentar minimizar la distorsión de fase, pero nunca logran alcanzar a los filtros FIR sin perder su capacidad de filtrado.
- Tienen un orden menor que los filtros FIR.
- Aun cuando los polos estén situados en el círculo de radio unidad pueden presentar inestabilidades si se implementan con aritmética de coma fija.

Existen dos tipos de filtros IIR: **AR** (Auto regresivo), y **ARMA** (Auto regresivo y media de movimiento).

Anexos

Características Filtros IIR Auto regresivos (AR)

- La ecuación que define a un filtro AR es:

$$x[n] = y[n] + A_1 y[n-1] + A_2 y[n-2] + \ldots + A_N y[n-N]$$

$$H[z] = \frac{1}{1 + A_1 z^{-1} + A_2 z^{-2} + \ldots + A_N z^{-N}}$$

- La función de transferencia contiene solo polos.
- El filtro es recursivo ya que la salida depende no solo de la entrada actual sino además de valores pasados de la salida (realimentación).
- El término auto-regresivo tiene un sentido estadístico en que la salida y[n] tiene una regresión hacia sus valores pasados.
- La respuesta el impulso es normalmente de duración infinita, de ahí su nombre.

Características Filtros IIR Auto regresivos y media de movimiento (ARMA)

- Es el filtro más general y es una combinación de los filtros MA y AR.
- La ecuación en diferencias que describe el filtro ARMA de orden N es:

$$y[n] + A_1 y[n-1] + A_2 y[n-2] + \ldots + A_N y[n-N] = B_0 x[n-1] + B_1 x[n-2] + \ldots + B_M x[n-N]$$

$$H[z] = \frac{B_0 + B_1 z^{-1} + B_2 z^{-2} + \ldots + B_M z^{-N}}{1 + A_1 z^{-1} + A_2 z^{-2} + \ldots + A_N z^{-N}}$$

- Un filtro de este tipo se denota por ARMA (N, M), cuando es **Auto-regresivo** de orden N y **Media de Movimiento** de orden M.
- Su respuesta al impulso es también de duración infinita y por tanto es otro tipo de filtro IIR.

Anexos

Métodos de diseño de filtros IIR:

Existen dos filosofías diferenciadas para el diseño de filtros IIR. De forma directa o indirecta.

Indirecta:

Se basa en aplicar a filtros analógicos diseñados previamente, transformaciones que los conviertan en digitales con las mismas características. Hay tres métodos fundamentales:

- Diseño por impulso invariante
- Diseño por analogía o aproximación de derivadas.
- Diseño por transformación bilineal.

Directa:

Se propone el diseño de filtros digitales imponiendo una serie de condiciones a la respuesta para determinar los coeficientes. Nos centramos en dos métodos simples como son:

- Diseño por la aproximación de Padé.
- Diseño por aproximación de mínimos cuadrados.
- También podemos considerar como método de diseño aunque de uso limitado la colocación de ceros y polos.

En nuestro experimento para el diseño de filtros nos ayudaremos con la herramienta de diseño de filtros de **MATLAB FDATool**, donde seleccionaremos los filtros **Elípticos (filtro de Cauer)** para el diseño de los distintos tipos de filtros IIR.

Condiciones de estabilidad:

Los filtros IIR más generales (ARMA) contienen ceros y polos. Si los coeficientes del filtro son reales, si los ceros o polos son complejos siempre aparecen como pares complejos conjugados.

La condición de estabilidad, para sistemas causales implica que los **POLOS se encuentran en el interior de la circunferencia unidad**. Los ceros no tienen efecto sobre la estabilidad del sistema y pueden encontrarse en el interior o en el exterior de dicha circunferencia.

Cuando los ceros y polos de un sistema se encuentran en el interior de la circunferencia unidad se dice que el sistema es de **FASE MÍNIMA**.

Cuando todos los ceros y polos están en el exterior de la circunferencia unidad se dice que el sistema es de **FASE MÁXIMA**.

En general, cuando tenemos ceros y polos en el exterior y en el interior se dice que el sistema es de **FASE MIXTA**.

Anexos

Tipos de Filtros:

Repetiremos a modos de repaso los cuatro tipos de filtros principales **tanto para FIR como para IIR** en cuanto al tratamiento de la señal de entrada:

-Los filtros pasa-bajos, pasa-altos, pasa-banda y los para-banda.

Filtro pasa-bajos:

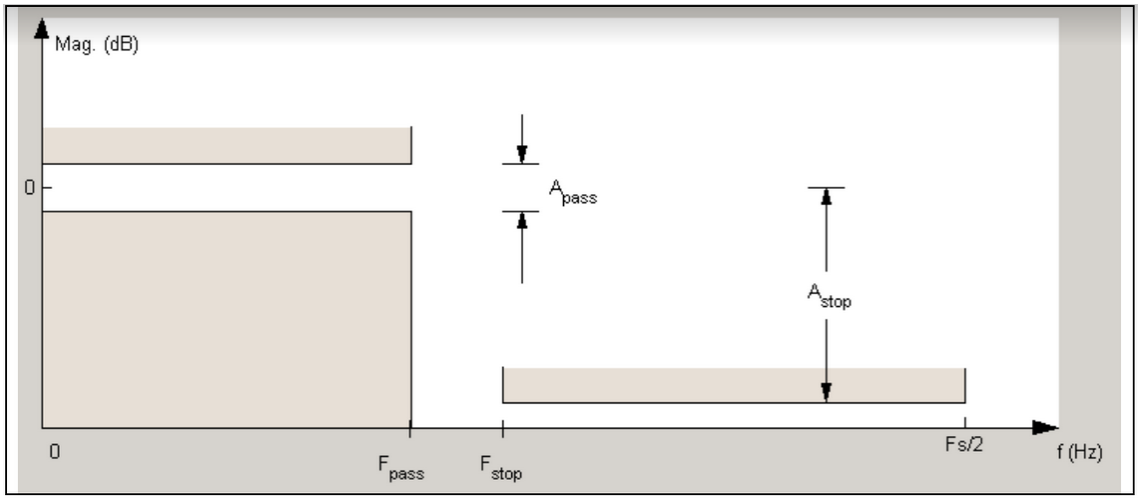


Ilustración 48: Filtro pasa-bajos

Donde *Apass* es el rizado de paso tolerable*, Astop* la atenuación en la banda de rechazo. Y en la banda de transición formada por la frecuencia de paso *Fpass* y la frecuencia de rechazo *Fstop*, denominada banda de transición.

Filtro pasa-altos:

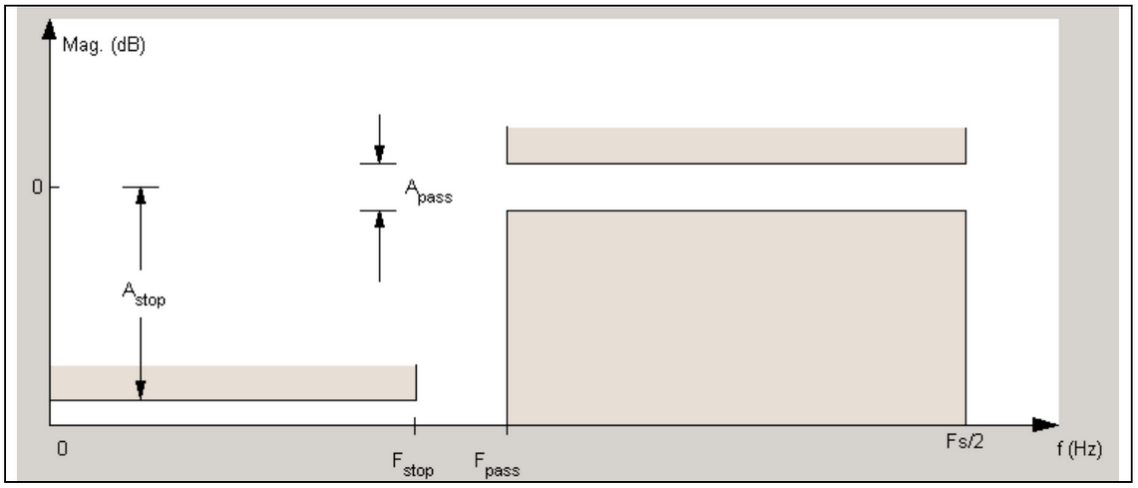


Ilustración 49: filtro pasa-altos

Aquí vemos a Fs/2 como la máxima frecuencia de paso debido al muestreo de la señal.

Anexos

Filtro pasa-banda:

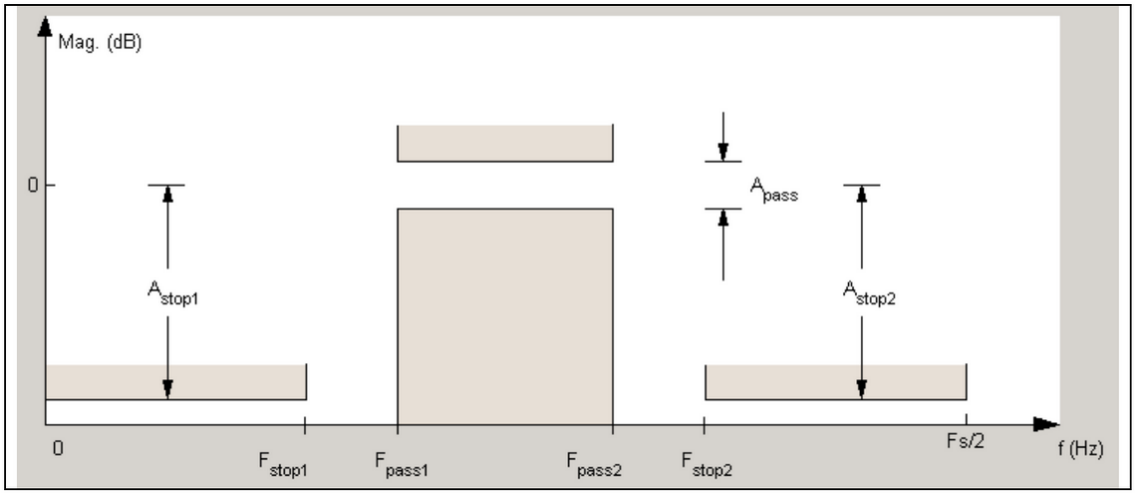


Ilustración 50: filtro pasa-banda

Filtro para-banda:

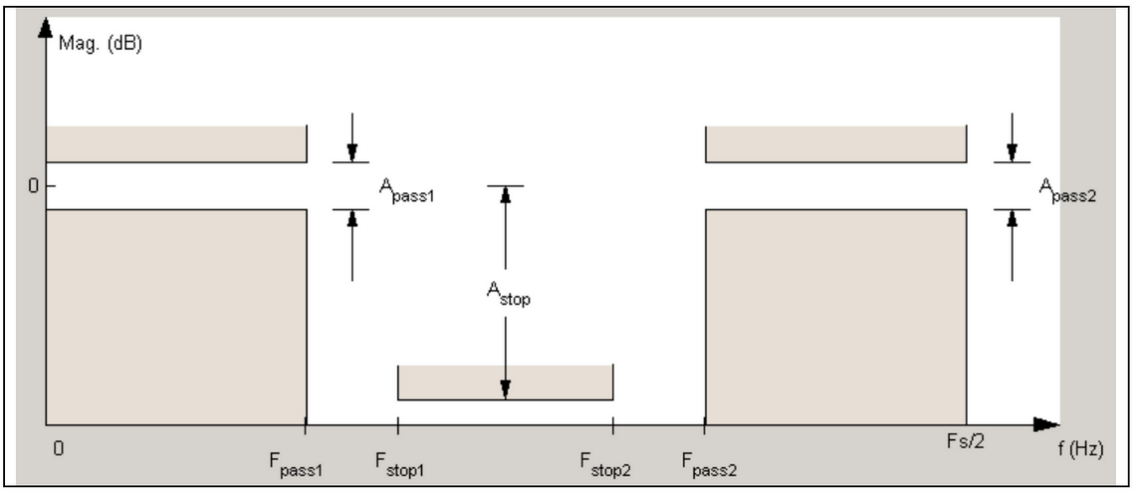


Ilustración 51: filtro para-banda

Anexos

Filtrado Digital de Señal Con DSP

Guía IIR

Iñigo Rodríguez Martínez

Tutor: Antonio Moisés Zorzano Martínez

31/01/2017

Procesador de señal a emplear: **eZdsp TMS320VC5505 USB Stick**

UNIVERSIDAD DE LA RIOJA

A lo largo de este documento se describen los pasos a seguir necesarios para la realización del experimento de procesado digital que desarrollara un filtrado IIR de principio a fin. Siendo este experimento real y realizándolo en nuestro DSP seleccionado.

Anexos

Contenido

Anexos

Introducción IIR:

Tras la lectura del documento de Información General IIR, en esta guía encontraremos los pasos a seguir para la utilización de todos los programas que se utilizarán en el desarrollo del filtrado IIR planteado como experimento.

El alumno en un primer momento deberá simplemente seguir los pasos descritos y limitarse a adquirir los conceptos y la metodología que se describe a continuación.

Si no se siguen los pasos se podrán dar errores de compilación u otros resultados finales que nos son los que busca presentar esta guía, por lo que se realizarán las pruebas una vez terminado el seguimiento de los pasos.

Recordamos que en la columna de filtrado IIR se recorrerá de arriba abajo, ya que los pasos siguen un orden lógico para conseguir el resultado deseado.

Selección tipo de filtro:

Como hemos visto en el último apartado de información general de filtrado IIR, encontramos diferentes tipos de filtros en función del tipo de tratamiento que queramos dar a la señal.

A diferencia de con el filtrado FIR, aquí tendremos un único script que se abrirá cuando cliquemos sobre "MATLAB-FDATool" Contendrá comentarios sobre el experimento y el comando necesario para abrir la Toolbox a emplear para el diseño de filtros.

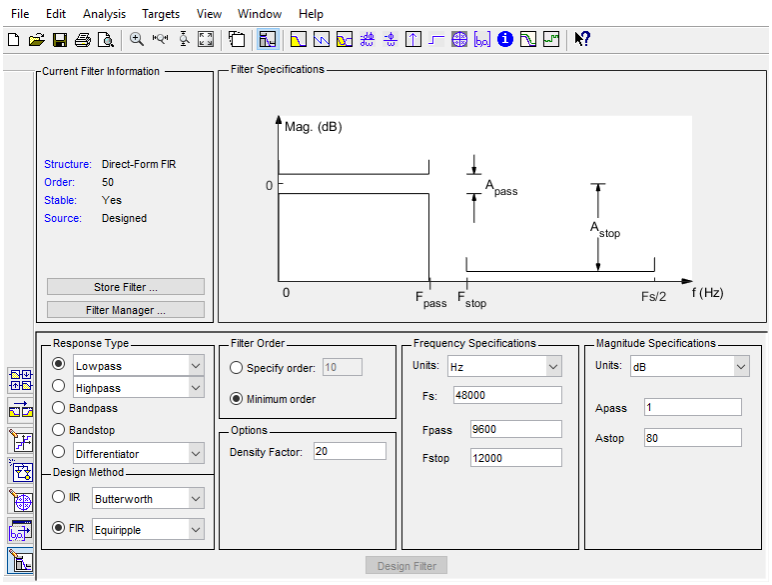
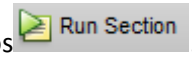


Ilustración 52: FDATool

Anexos

Pasos para realizar el primer bloque:

En primer lugar comprobaremos que el workspace de MATLAB se encuentra en la carpeta de archivos de la herramienta, en el apartado de los archivos de Matlab, y en concreto en la carpeta destinada al filtrado FIR.

Tras esto leemos los comentarios del script y lo ejecutamos 

. En ese momento se nos abrirá una ventana que como hemos visto en la imagen anterior se tratará de la toolbox FDATool a partir de aquí seguiremos los siguientes pasos para el diseño del filtro:

1. Pulsaremos sobre el icono de abrir  en la ventana de FDATool donde encontraremos distintas sesiones guardadas con cada uno de los tipos de filtro con extensión (**.fda**), donde seleccionamos **filtrado Pasa-banda** ya que es el más representativo a nivel visual de la respuesta. Cuando clicamos se cargará la sesión con un filtro ya diseñado.

2. Recorreremos detenidamente las diferentes pestañas para el diseño del filtro que encontraremos a la izquierda analizando el contenido de las mismas.
   Lo pestaña principal será la última donde están introducidos el tipo de filtro, **IIR elíptico**, la frecuencia de muestreo (**8000 HZ**), las frecuencias de paso y de stop, y las distintas atenuaciones.

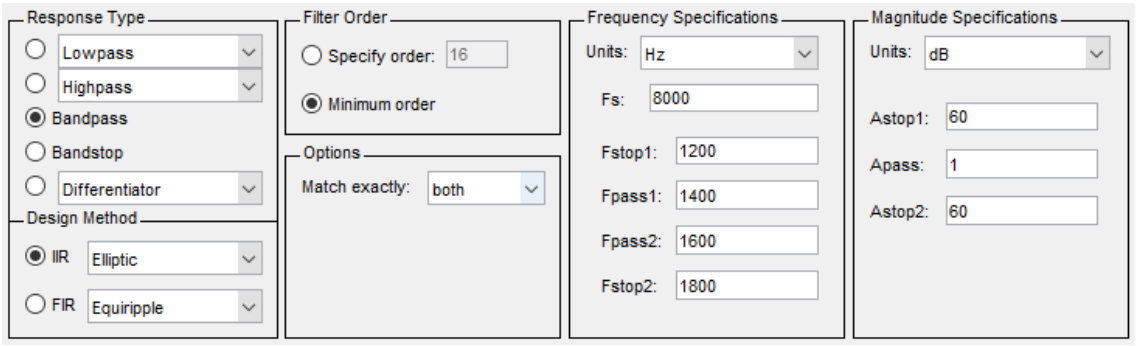


Ilustración 53: Configuración FDATool

   En la primera pestaña veremos que hemos utilizado un modelo matemático de integrador en cascada **"Cascaded Integrator Comb"**, y en la tercera pestaña hemos seleccionado la aritmética del filtro en coma fija, **"Fixed Point".**

3. Tras comprobar que todo lo descrito en el paso dos esta seleccionado, volveremos a la última pestaña de diseño y pulsaremos en 

, para que calcule los coeficientes y nos muestree en la imagen la forma del filtro.
   Si clicamos en: **Analysis-> Magnitude Response Estimate** nos cambiará la imagen representada en la FDATool para mostrarnos una estimación de la respuesta en magnitud en línea continua, y manteniendo la forma anterior de los coeficientes en línea discontinua.

Anexos

En el panel de la izquierda vemos que el orden obtenido de nuestro filtro IIR es ocho, mucho menor que el obtenido en el experimento FIR.

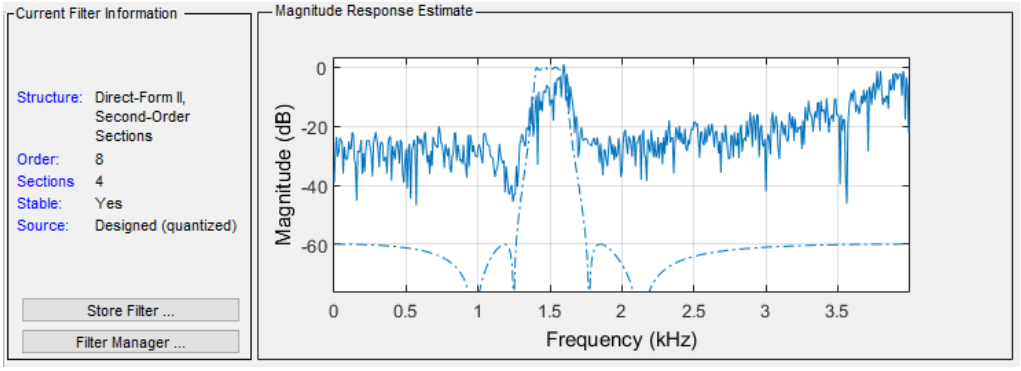


Ilustración 54: Representación FDATool

Analizando el resultado vemos claramente que no se trata de un filtro ideal, sino que posee un rizado y unas bandas de transición donde el resultado dependerá de cómo de cerca o de lejos nos encontremos de las frecuencias introducidas ya que estas sí que están bastante bien fijadas y con un valor en magnitud adecuado.

4. Para exportar los coeficientes obtenidos en el filtro en un formato que podamos utilizar en **CCS**, Matlab nos ofrece una compatibilidad directa entre esta Toolbox y la aplicación CCS, así que haremos uso de su propio conversor. Clicaremos en la barra de menús en :

   Targets -> Code Composer Studio (tm) IDE

   Se nos abrirá un menú para exportar el archivo donde lo único que cambiaremos será el tipo de dato a exportar que seleccionaremos: **signed 16 bits integer**

   Respetaremos los nombres de las variables por defecto lo que nos generará un archivo en el que no tendremos que tratar los coeficientes para poder introducirlos.

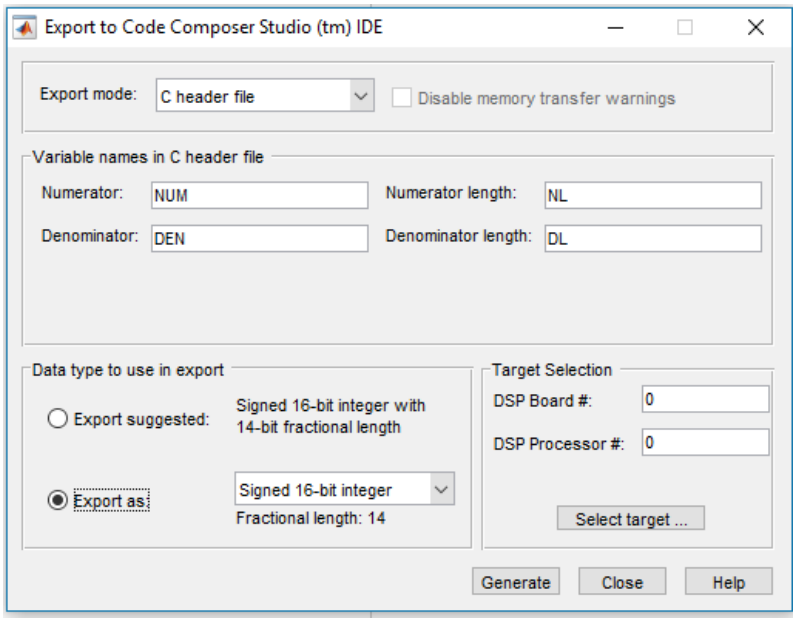


Ilustración 55: Exportar filtro

Pulsaremos en “Generate” y nos abrirá un dialogo para ver donde queremos guardar el archivo. Lo guardaremos en la carpeta de la herramienta, donde se encuentran los archivos de Matlab, relativos al filtrado IIR, llamada: “Coeficientes Generados”, con un nombre nuevo donde no borremos ningún archivo de los existentes.

5. Cerramos Matlab por completo, y volvemos a la herramienta. Y pulsaremos en el siguiente apartado, “Coeficientes Generados” donde se nos abrirá la carpeta donde hemos guardado los coeficientes generados.

   Clicamos en el siguiente apartado del programa sin cerrar el anterior, “Coeficientes a utilizar” y se nos abrirá la carpeta donde se encuentran los coeficientes en el experimento. Para sustituirlos, bastará con coger el archivo de coeficientes generados, copiarlo en la carpeta de los que utilizamos sustituyéndolo por el que ya había dejándolo con el mismo nombre.

   El nombre del archivo a sustituir es: **fdacoefsMATLAB.h**

   También tenemos otra opción más costosa que es abrir los dos documentos como block de notas y copiar los coeficientes generados en el archivo **fdacoefsMATLAB.h**

Con esto habremos terminado el apartado de selección y diseño del filtro, habiendo introducido los coeficientes en el programa de filtrado del experimento IIR.

Anexos

Filtrado mediante DSP:

Llegados a este punto accedemos al siguiente cuadro avanzando así con el experimento. Ya tenemos diseñado nuestro filtro.

Pasos para realizar el segundo bloque:

Al igual que en el experimento relativo al filtrado FIR, también trataremos señales de audio, por lo que aunque el programa empleado sea distinto, desde un punto de vista procedimental para usarlo, los pasos a seguir serán similares.

1. Pulsaremos en el siguiente botón "Selección de entrada" donde se nos abrirá un carpeta que contiene señales de entrada de audio generadas por la SoundCard Scope, que podremos introducir. Sin cerrar este pulsamos en el botón introducir entrada y se nos abrirá la carpeta donde se encuentra introducida la señal a procesar en el experimento en formato .wav.

2. Cogeremos la señal de la carpeta de entradas llamada "Default" y la copiaremos en la donde introduciremos la entrada, sustituyendo su nombre por **in.wav** y borrando la que había en la carpeta de destino con el mismo nombre.

3. En este punto ya tenemos todas las entradas listas, por lo que procedemos a conectar el Hardware necesario: **eZdsp TMS320VC5505 USB Stick** en una de las entradas usb disponibles en el pc.

4. Clicamos en el botón "Ejecutar EzDSP" aquí se nos abrirá CCS (Code Composer Studio), que nos hará seleccionar un directorio de trabajo, donde seleccionaremos en la carpeta de los archivos de la herramienta la llamada "**Experimento IIR**" como Workspace.

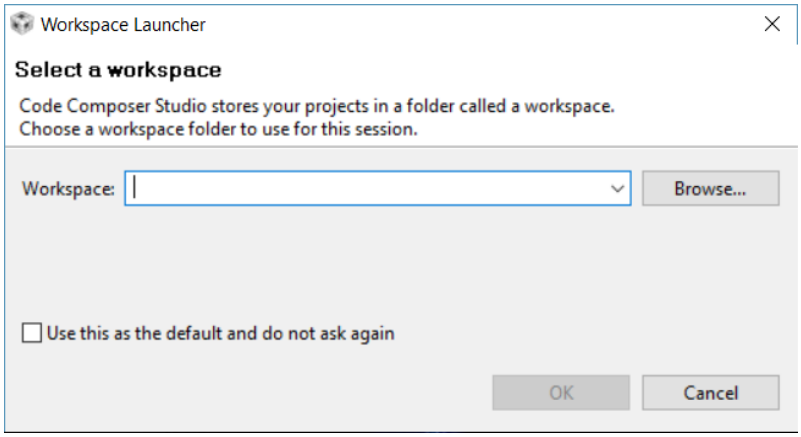
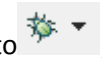


Ilustración 56: Selección workspace

(…/Archivos_TFG\Experimentos\Experimento_IIR)

Anexos

5. El experimento está ya preparado, por lo que intentaremos comprender lo que se realiza en él, en los tres archivos que nos ha abierto:
   El primero corresponde al procesado de la señal **fixPoint_cascadeIIR.c**, el segundo es el que nos enlazará las distintas secciones pidiendo la introducción de comandos por pantalla para la realización del experimento **fixPoint_cascadeIIRTest.c**, y el tercero son los coeficientes calculados que ya habíamos introducido **fdacoefsMATLAB.h.**

6. Ahora compilaremos y ejecutamos el proyecto  , compilara y si no hay ningún problema de compilación ni warnings ni errores nos abrirá la ventana de depuración.

7. Donde para ejecutar, bastará con darle al **"play"**, tras esto atenderemos a la ventana de comandos donde nos pedirá que introduzcamos un número para comenzar el experimento. Tras esto esperaremos hasta que nos devuelva que el experimento ha sido completado.
   Para terminar la ejecución bastará con pulsar el botón de **Stop.**

8. Cerraremos CCS y clicamos en la siguiente opción de nuestra herramienta "Comprobación de salida" que nos abrirá la carpeta con la señal de salida procesada y podremos escucharla para comprobar su tratamiento.
   El nombre de la señal obtenida es: **out.wav**

Anexos

Comprobación de la salida:

Tras la comparación de salida auditiva donde notaremos diferencia con la señal de entrada dado que en el experimento realizado eliminaremos dos de las senoidales, de las tres que se componía la señal de entrada inicial.

Pasaremos ahora de realizar una comprobación más exhaustiva, donde podremos ver como se ha filtrado y por tanto nos dará una mayor cantidad de datos y un mejor tipo de análisis ya que de oído no podemos diferenciar las frecuencias filtradas o las que no.

Pulsaremos en el siguiente apartado "Comprobación MATLAB", este nos devolverá al programa **MATLAB**, donde nos abrirá otro script ya implementado para la lectura de archivos **.wav**, y la visualización de los resultados del filtrado.

Pasos para realizar el tercer bloque:

1. Para que el script funcione correctamente, hemos de colocar el workspace de Matlab en la carpeta donde se encuentra el script, que se encuentra en la misma carpeta donde depositamos las señales de audio en la carpeta de experimento de CCS.
   …/Archivos_TFG\Experimentos\Experimento_IIR\EXP_fixed_point_fir\data
2. A continuación realizaremos una lectura del script intentando comprender los comandos introducidos y buscando en la ayuda lo que no entendemos.
3. Ejecutaremos el script sección a sección analizando los resultados en el apartado de variables de MATLAB, comprendiendo los valores obtenidos en función de las señales introducidas.
4. Por ultimo al ejecutar la última sección del script, se nos abrirá un gráfico con la señal sin filtrar "original" y la señal tratada "filtrada" donde podremos comprobar el trabajo del filtro y su correcto funcionamiento.

Anexos

Apartado de aprendizaje Autónomo:

Una vez terminada la realización y cogido la metodología empleada en el segundo experimento, volveremos al principio del bloque de "Filtrado IIR" y volveremos a recorrer los distintos pasos de la herramienta, seleccionando otro tipo de filtro, modificando las frecuencias reseñables, para modificar donde se realizarán los cambios en el filtrado.

También tendremos la oportunidad de cambiar la señal de entrada, con otras frecuencias, tomando las generadas con la **SoundCard Scope**, en el ejercicio del primer bloque o generar una señal en el momento para que corresponda con las frecuencias de filtrado introducidas en el filtro.

Cuando ejecutemos todos los pasos del filtrado obtendremos los resultados acordes al filtrado realizado y la señal introducida obteniendo una señal nueva fruto del procesado real de la señal en el DSP en **Code Compose Studio**.

A la cual podremos analizar con el script de **MATLAB** de la misma manera con la que se comprobó la señal de salida del experimento guiado.

Nótese que si realizamos por ejemplo un filtro pasa-bajos a una frecuencia introducida por el alumno y la señal de entrada a filtrar solo se compone de una frecuencia en la que el filtro deja pasar la señal, el filtro como es evidente no realizará ninguna operación. Y por lo tanto obtendremos una señal de salida idéntica a la de entrada.

Anexos

Filtrado Digital de Señal Con DSP

Ejercicios IIR

Iñigo Rodríguez Martínez

Tutor: Antonio Moisés Zorzano Martínez

31/01/2017

Procesador de señal a emplear: **eZdsp TMS320VC5505 USB Stick**

UNIVERSIDAD DE LA RIOJA

A lo largo de este documento se plantean ejercicios a desarrollar por el alumno para continuar con el aprendizaje del procesado digital, no limitándonos solo a los experimentos realizados y actividades guidas. Ya que se asentarán mejor los conocimientos y se adquirirán nuevos poniéndolo en práctica.

Anexos

Introducción:

Se plantean en este documento una serie de ejercicios más o menos abiertos en función de los casos, que buscan un mayor manejo de la herramienta y la continuación del aprendizaje tras la parte guiada de los experimentos.

Se recomienda la lectura total de cada ejercicio antes de empezar a realizarlo ya que muchos de los objetivos pueden realizarse de manera simultánea.

Ejercicios planteados:

Utilizando la FDATool, se plantean diferentes filtros para probar en el experimento anterior.

Se guardaran las sesiones de la FDATool y se incluirán capturas de pantalla de los diseños.

1. Genera un filtro Butterworth, que realice un filtrado pasa-altos, con frecuencia de paso 2500 Hz, y frecuencia de muestreo 48000 Hz.
   Modifica de tal manera el filtro con un mínimo de atenuación en la banda de stop de 30 dB, y una frecuencia de stop lo más cercana a la frecuencia de paso.
   Cabe recordar que los filtros IIR que presentan gran número de coeficientes tienen una mayor facilidad de volverse inestables, además de que su coste computacional aumenta considerablemente.
   Por lo que la última restricción es conseguir que el orden del filtro este por debajo de 20.

2. Genera un filtro Chevyshev tipo 1. Que realice un filtrado pasa-bajos, con frecuencia de paso 1200 Hz, y frecuencia de muestreo de 48000 Hz.
   Las condiciones pedidas son que el orden del filtro sea menor a 10, con una atenuación en la banda de stop mayor de 60 dB, y la frecuencia de stop lo más cercana posible a la frecuencia de paso

3. Genera un filtro Chevyshev tipo 1. Que realice un filtrado pasa-banda, con frecuencias en la banda de paso 1400 y 2200 Hz, y frecuencia de muestreo de 48000 Hz.
   Las condiciones pedidas son que el orden del filtro sea menor a 16 con una atenuación en las bandas de stop de 60 dB, seleccionando las frecuencias de stop lo más cercanas posibles a las frecuencias de paso.
   -Comprueba que sucede al aumentar el parámetro *Apass*, y que produce en la forma del filtro.

Anexos

4. Genera un filtro Elíptico, que realice un filtrado para-banda, con frecuencias de paso de 2500 y 3400 Hz y una frecuencia de muestreo de 48000 Hz.
   Las condiciones pedidas son que el orden del filtro sea menor a 8, con una atenuación en la banda de stop de 60 dB, y unas atenuaciones en la banda de paso de 1.
   Selecciona las frecuencias de stop lo más cercanas posibles a las frecuencias de paso que cumplan estas características.

Anexos

Filtrado Digital de Señal Con DSP

Código reseñable del experimento

Iñigo Rodríguez Martínez

Tutor: Antonio Moisés Zorzano Martínez

31/01/2017

Procesador de señal a emplear: **eZdsp TMS320VC5505 USB Stick**

UNIVERSIDAD DE LA RIOJA

En este documento se introducirá el código reseñable de los programas que merezca la pena destacar de tal manera que queden reflejados en el documento.

Anexos

Código CCS:

- Archivo coeficientes del filtro:

```
#define MWSPT_NSEC 9
const int NL[MWSPT_NSEC] = { 1,3,1,3,1,3,1,3,1 };
const int16_T NUM[MWSPT_NSEC][3] = {
  {
     1491,      0,      0
  },
  {
    16384, -23636,  16384
  },
  {
     1491,      0,      0
  },
  {
    16384,   3306,  16384
  },
  {
     6214,      0,      0
  },
  {
    16384, -18247,  16384
  },
  {
     6214,      0,      0
  },
  {
    16384,  -5864,  16384
  },
  {
    16384,      0,      0
  }
};
```

```
const int DL[MWSPT_NSEC] = { 1,3,1,3,1,3,1,3,1 };
const int16_T DEN[MWSPT_NSEC][3] = {
  {
    16384,      0,      0
  },
  {
    16384, -13231,  15528
  },
  {
    16384,      0,      0
  },
  {
    16384, -11232,  15504
  },
  {
    16384,      0,      0
  },
  {
    16384, -14706,  16069
  },
  {
    16384,      0,      0
  },
  {
    16384, -10054,  16049
  },
  {
    16384,      0,      0
  }
};
```

Ilustración 57: Coeficientes IIR

Anexos

- Código principal experimento:

```
#include <stdio.h>
#include <stdlib.h>
#include "tistdtypes.h"
#include "cascadeIIR.h"
#include "fdacoefsMATLAB.h"

void wHdUpdt(Uint8 *w, Uint32 bytes);

#define SECTIONS        ((MWSPT_NSEC-1)/2)      // Number of 2nd order sections
Int16   C[SECTIONS*5]; // Filter coefficients obtained from example 5.14 MATLAB FDATool
                       // C[]=A[i][1], A[i][2], B[i][2], B[i][0], B[i][1]...
Int16   w[SECTIONS*2]; // Filter delay line
                       // w[]=w[i][n-1],w[i+1][n-1],...,w[i][n-2],w[i+1][n-2],...

#define NUM_DATA        160                     // Number of samples per block
Int16   out[NUM_DATA]; // Filter output data buffer
Int16   in[NUM_DATA];  // Filter input data buffer


void main(void)
{
    Int16 i,k,n,c;
    Int16 gainNUM,gainDEN;
    Int32 temp32;
    Int8  temp[NUM_DATA*2];
    FILE  *fpIn,*fpOut;
    Uint8 waveHeader[44];
    Uint32 cnt;

    printf("Enter 1 for using WAV file and START EXPERIMENT\n");
    scanf ("%d", &c);

    if (c == 1)
    {
        fpIn = fopen("..\\data\\in.wav", "rb");
        fpOut = fopen("..\\data\\out.wav", "wb");
    }

    else
    {
        printf("Cancelled, Restart Experiment");
        exit(0);
    }
    // Open file for read input data
    if (fpIn == NULL)
    {
        printf("Can't open input data file\n");
        exit(0);
    }

    if (c == 1)                                 // Create WAVE data file header
    {
        fread(waveHeader, sizeof(Int8), 44, fpIn);
        fwrite(waveHeader, sizeof(Int8), 44, fpOut);
    }

    // Initialize IIR filter delay line
    for (i=0; i<SECTIONS*2;i++)
    {
        w[i] = 0;
    }
    cnt = 0;
```

Ilustración 58: Programa principal CCS IIR

Anexos

```
95      printf("Exp --- IIR filter experiment\n");
96
97      // IIR filter experiment start
98      while (fread(temp, sizeof(Int8), NUM_DATA*2, fpIn) == (NUM_DATA*2))
99      {
100             for (k=0, i=0; i<NUM_DATA; i++)
101             {
102                 in[i] = (temp[k]&0xFF)|(temp[k+1]<<8);
103                 k += 2;
104             }
105
106             cascadeIIR(in,NUM_DATA,out,C,SECTIONS,w); // Filter a block of samples
107
108             for (k=0, i=0; i<NUM_DATA; i++)
109             {
110                 temp[k++] = (out[i]&0xFF);
111                 temp[k++] = (out[i]>>8)&0xFF;
112                 cnt += 2;
113             }
114             fwrite(temp, sizeof(Int8), NUM_DATA*2, fpOut);
115     }
116
117     if (c == 1)     // Based on input WAVE file to generate output
118     {
119         wHdUpdt(waveHeader, cnt);
120         rewind(fpOut);
121         fwrite(waveHeader, sizeof(Int8), 44, fpOut);
122     }
123
124     fclose(fpIn);
125     fclose(fpOut);
126     printf("Exp --- completed\n");
```

- Código añadido para filtrado:

```
14  #include "tistdtypes.h"
15
16  void cascadeIIR(Int16 *x, Int16 Nx, Int16 *y, Int16 *coef, Int16 Ns, Int16 *w)
17  {
18      Int16 i,j,n,m,l,s;
19      Int16 temp16;
20      Int32 w_0, temp32;
21
22      m=Ns*5;                             // Setup for circular buffer coef[]
23      s=Ns*2;                             // Setup for circular buffer w[]
24
25      for (j=0,l=0,n=0; n<Nx; n++)        // IIR filter begin
26      {
27          w_0 = (Int32)x[n]<<12;          // Scale input to prevent overflow
28          for (i=0; i<Ns; i++)
29          {
30              temp32 = (Int32)(*(w+l)) * *(coef+j); j++; l=(l+Ns)%s;
31              w_0   -= temp32<<1;
32              temp32 = (Int32)(*(w+l)) * *(coef+j); j++;
33              w_0   -= temp32<<1;
34              w_0   += 0x4000;            // Rounding
35
36              temp16 = *(w+l);
37              *(w+l) = (Int16)(w_0>>15); // Save in Q15
38
39              w_0    = (Int32)temp16 * *(coef+j); j++;
40              w_0  <<= 1;
41              temp32 = (Int32)*(w+l) * *(coef+j); j++; l=(l+Ns)%s;
42              w_0   += temp32<<1;
43              temp32 = (Int32)*(w+l) * *(coef+j); j=(j+1)%m; l=(l+1)%s;
44              w_0   += temp32<<1;
45              w_0   += 0x800;             // Rounding
46          }
47          y[n] = (Int16)(w_0>>12);        // Output in Q15 format
48      }
```

Ilustración 59: Código filtrado IIR

Anexos

Código Matlab:

- Código de comprobación

```
%% Lectura de las señales de audio in/out
[a, Fsr]=audioread('input.wav');
L=length(a);

[a1, Fsr1]=audioread('output.wav');
L1=length(a1);
%% Tratamiento para su representación
NFFT= 2^nextpow2(L);
Y=fft(a,NFFT)/L;
f=Fsr/2*linspace(0,1,NFFT/2+1);

NFFT1= 2^nextpow2(L1);
Y1=fft(a1,NFFT1)/L1;
f1=Fsr1/2*linspace(0,1,NFFT1/2+1);
%% Muestreo de las señales
subplot(2,1,1)
plot(f, 2*abs(Y(1:NFFT/2+1)))
legend ('Señal original')
subplot(2,1,2)
plot(f1, 2*abs(Y1(1:NFFT1/2+1)),'r') %señal en frecuencia
legend ('Señal filtrada')
```

Ilustración 60: Código comprobación Matlab

Anexos

Anexo 4:

Filtrado Digital de Señal Con DSP

Comparación de Filtros

Iñigo Rodríguez Martínez

Tutor: Antonio Moisés Zorzano Martínez

31/01/2017

Procesador de señal a emplear: **eZdsp TMS320VC5505 USB Stick**

UNIVERSIDAD DE LA RIOJA

A lo largo de este documento se ofrece una visión general parte por parte del experimento a realizar, finalidad del mismo y conclusiones que se deberán sacar sobre la comparación de los distintos métodos de filtrado.

Anexos

Contenido

Anexos

Introducción:

Tras la realización de los experimentos anteriores, deberíamos saber diseñar siguiendo el método utilizado tanto filtros FIR como IIR, en mayor medida el filtrado FIR mediante el método de Parks-McClellan y el filtrado IIR con el método de diseño de filtros Elípticos (Cauer), además de haber investigado como realizar otros métodos para la resolución de los ejercicios planteados.

En este experimento, aprenderemos a diseñar tanto filtros FIR como IIR mediante otros métodos de diseño, empleando para ello la herramienta de diseño de filtros FDATool, que ya hemos utilizado en el bloque de filtrado IIR.

En este caso, todos los filtros diseñados serán pasa-bajos, lo que no supondrá un problema en el aprendizaje ya que los otros tipos se realizarán de forma análoga.

Y además, pondremos a prueba los filtros generados y este es el porqué de que todos los filtros serán del mismo tipo, ya que se simulará el filtrado de una señal generada por nosotros mismo, para todos y cada uno de los filtros creados, generando una gráfica comparativa con las respuestas obtenidas.

Este experimento tiene una finalidad clara, que es la toma de decisiones por parte del alumno de que método de filtrado elegir en función de la señal a tratar, sacando las conclusiones pertinentes de los resultados de todos ellos de forma experimental.

Recordamos que en la columna de Comparación de filtros se recorrerá de arriba abajo, ya que los pasos siguen un orden lógico para conseguir el resultado deseado.

Anexos

Diseño de los Filtros:

Como hemos mencionado anteriormente el diseño de los filtros se realizara mediante la **FDATool**, utilizando su entorno de desarrollo para la generación de los filtros.

Se diseñaran **diez filtros** de los cuales la mitad serán del tipo FIR y la mitad restante del tipo IIR.

Los métodos de diseño elegidos han sido los más intuitivos, siendo más fáciles de ver en la gráfica de representación de la herramienta el procedimiento de filtrado.

Todos los filtros corresponderán a filtros "**Pasa-bajos**", siendo la frecuencia de paso 2000 Hz y la frecuencia de paro o de stop 2500 Hz. Se ha intentado que la atenuación en general sea de 60 dB, pero no es posible introducir esta atenuación en todos los filtros, sobre todo en los filtros IIR donde queremos el menor número de coeficientes posibles y en aquellos métodos que presentan pendientes más suaves. Esta especificación aumenta drásticamente el número de coeficientes perdiendo el filtro utilidad y presentando inestabilidades.

Cabe recordar que los filtros IIR pueden volverse inestables, por lo que en el apartado de aprendizaje autónomo podremos obtener resultados inesperados por parte del alumno. Se analizaran estos casos y se verá por qué se producen.

Generación de la señal a filtrar:

Todos los apartados siguientes se realizarán en MATLAB, y estarán integrados en el mismo "script" por lo que como hemos aprendido como ejecutar secciones de código paso a paso, llevaremos a cabo este procedimiento para facilitar el aprendizaje.

La señal generada será una senoidal, con parámetros modificables como: la frecuencia, la amplitud y la frecuencia de muestreo utilizada en su creación.

Además, se generará una señal de ruido que no emplearemos en primeras instancias, pero que, cuando la sumemos, obtendremos una señal más realista, pudiendo obtener así unos resultados de la simulación más próximos a la realidad.

Se muestreará la señal creada para poder ver en todo momento que es lo que estamos haciendo y aclarando así conceptos que de otra manera pudiesen quedar sin resolver.

Con el uso de Matlab en la generación se tiene como objetivo indirecto la toma de cercanía con el desarrollo de código en este entorno de programación, lo cual nos será útil en muchos ámbitos.

Anexos

```
%% Generación de la señal a procesar
Fs = 24000;      % Frecuencia de muestreo
Ts = 1/Fs;       % Numero de muestras
F = 2000;        % Frecuencia de la senoidal
Amp = 5;         % Amplitud de la senoidal
sinewave = Amp*sin(2*pi*F*(0:Ts:1));          % Generación de la senoidal
noise = sqrt(0.1).*randn(1,length(0:Ts:1));   % var = 0.1 Ruido blanco
xn = sinewave%+noise;                % Suma del ruido a la señal
xn_int = xn(1:256);                  % Nos quedamos con las 256 primeras muestras
plot(xn_int); axis([1 256 -Amp Amp]); % Visualizamos
ylabel('Amplitude'); xlabel('Sample index, n'); % Etiquetas gráfico
```

Ilustración 61: Generación de la señal

Filtrado de la señal generada con los diversos filtros:

A continuación en las siguientes dos secciones se realizará el filtrado de la señal generada. Para ello, se exportarán los coeficientes de la FDATool al workspace de Matlab de una manera concreta que se presentará en el documento guía. Se dedicarán las dos secciones siguientes, una al filtrado con los filtros FIR, y otra para el filtrado con los filtros IIR. Se han diferenciado debido a que el formato de los coeficientes adquiridos es distinto y se realizará un pretratamiento para dejarlos todos de la misma manera.

Por último, se filtrará la señal diez veces, una con cada filtro.

Representación de los resultados

Tras la obtención de los diferentes señales resultantes del filtrado, se representarán en dos gráficas para no emborronar demasiado en una sola, divididos en dos bloques los filtros FIR y los filtros IIR, una sobre la otra, pudiendo comparar así los diferentes tipos en cuanto a sus resultados. Se podrá ver cuales han realizado un mejor tratamiento y cuales no son indicados para el tipo de filtrado empleado.

Tras la lectura de este documento seguiremos los pasos de la guía para obtener los resultados aquí descritos.

Anexos

Filtrado Digital de Señal Con DSP

Guía: Comparación de Filtros

Iñigo Rodríguez Martínez

Tutor: Antonio Moisés Zorzano Martínez

31/01/2017

Procesador de señal a emplear: **eZdsp TMS320VC5505 USB Stick**

UNIVERSIDAD DE LA RIOJA

A lo largo de este documento se describen los pasos a seguir necesarios para la realización del experimento de procesado digital que desarrollará diez de los filtros más comunes y sometiéndolos al filtrado de la misma señal para comprobar la eficacia y poder compararlos. A diferencia de los otros experimentos, éste será una simulación y se realizará enteramente en MATLAB y la Toolbox de diseño de filtros FdaTool.

Anexos

Contenido

Anexos

Introducción:

En este documento se presentan los pasos a seguir para la realización completa del experimento dedicado a la comparación de filtros con MATLAB.

El alumno en un primer momento deberá simplemente seguir los pasos descritos, limitándose a adquirir los conceptos y la metodología que se describe a continuación.

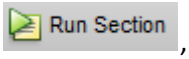
Si no se siguen los pasos, se podrán dar errores de compilación u otros resultados finales que nos son los que busca presentar esta guía, por lo que se realizarán las pruebas una vez terminado el seguimiento de los pasos.

Generación de los filtros:

Como se ha comentado en el documento de información general, todo el experimento se llevara a cabo en Matlab y en su toolbox de diseño de filtros. Debido a esto algunos aspectos serán más sencillos ya que dentro del programa todos los formatos son compatibles. Por ejemplo, no habrá que realizar copia uno a uno de los coeficientes.

Pasos Primer bloque:

1. En primer lugar desde la herramienta pulsamos sobre el botón “Comparación de filtros en MATLAB”. Éste nos abrirá MATLAB, con un script cargado que contendrá en distintas secciones el código a ejecutar en cada caso para realizar el experimento.

2. Leeremos la primera sección de código y la ejecutaremos de forma independiente 

, se nos abrirá la FDATool.
Dado que se trata del experimento guiado dentro de la Toolbox podremos pulsar en abrir sesión  , y encontraremos las diez sesiones guardadas de los diez tipos de filtros a comparar.

3. Abriremos el primero, ya que los comprobaremos de forma ordenada. Están numerados para evitar equivocaciones a la hora de exportar los coeficientes desde la Toolbox al Workspace de Matlab.

4. Comprobaremos que todos ellos tienen la frecuencia de paso en 2 KHz y la frecuencia de stop en los que podamos elegirla de 2,5 KHz. La frecuencia de muestreo en todos los casos será de 24 KHz.

Anexos

5. Tras esto procederemos a exportar los coeficientes al Workspace de Matlab, pulsando en: **File-> Export…** Esto nos abrirá una ventana para poder exportarlos.
   Tendrá dos formas posibles en función del tipo de filtro que estemos exportando, los filtros FIR tendrán un único coeficiente que por defecto será "Num", y los filtros IIR tendrán dos coeficientes que por defecto serán "SOS" y "G".

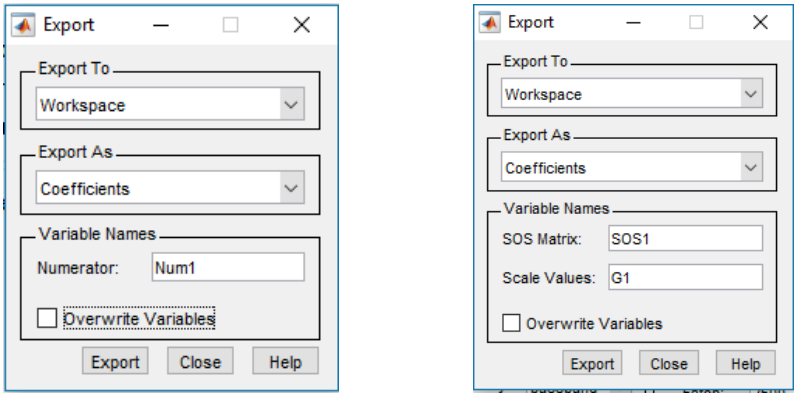


Ilustración 62: Exportar Matlab

En función de que sesión nos encontremos enumeraremos añadiendo el número correspondiente pegado al nombre por defecto de la variable a exportar del uno al cinco (1…5) los diferentes filtros. Por lo que el filtro **FIR** correspondiente a la sesión **uno** FIR tendrá su variable seguida de un **uno**, y la sesión **IIR** correspondiente a la sesión **seis** IIR, tendrá sus dos coeficientes seguidos de un **uno**.

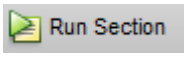
6. Realizaremos la misma operación con todas las sesiones creadas, comprobando su correcto diseño y exportando los coeficientes con los números añadidos correspondientes.

Este paso es crítico debido a que la representación y el filtrado están preparados para funcionar con esos nombres y con esas sesiones. Realizándolo de otra manera, no funcionaría el script, y si no lo realizamos en el orden sugerido, no corresponderá el filtro con la leyenda incluida en la posterior representación.

Anexos

Pasos Segundo bloque:

1. En este bloque se generará una señal senoidal, que será la señal de entrada a filtrar por los filtros obtenidos en los pasos anteriores.
   En primer lugar, leeremos detenidamente la siguiente script tratando de comprender su funcionamiento con ayuda de los comentarios incluidos, y en el caso de encontrar dudas no resultas en los comentarios, podremos utilizar la ayuda de Matlab donde simplemente escribiendo en la "Comand Window" help y el nombre de la función que no entendemos podremos acceder a la ayuda de Matlab.

2. Tras entender los comandos incluidos en el script, procederemos a ejecutar la segunda sección, pulsamos en 

   para ejecutarla.

3. Cuando termine de ejecutarlo nos realizará una representación de la señal generada, y nos guardará la señal de entrada tanto en el workspace como en un block de notas en el directorio de trabajo.

Filtrado de la señal:

Llegados a este punto, procederemos a pasar la señal generada en el apartado anterior por todos los filtros que hemos creado.

Pasos tercer bloque:

1. En la siguientes dos secciones de código del script se encuentran los comandos necesarios para realizar los diferentes filtrados. El primer bloque corresponde a los filtros FIR, y el segundo bloque a los filtros IIR.

2. Realizaremos una lectura de las dos secciones viendo cómo se realiza. En la primera sección, que coeficientes se introducen para realizar la simulación de filtrado. Además, si escribirnos en la "Comand Window" **help filter** encontraremos información adicional de cómo se realiza la simulación.
   La segunda sección IIR, tendrá la misma forma que la primera pero con un comando antes de cada filtrado, con el que transformaremos la forma en la que nos dan los coeficientes del filtro de tal manera que se pueda realizar el filtrado de la misma forma que con los filtros FIR.

3. Ejecutaremos las dos secciones y comprobaremos que en el workspace tenemos guardadas las salidas de los 10 filtrados con nombres de y1. a y10

Anexos

Comparación de filtros:

Una vez realizados todos los filtrados y obtenidos todas las señales resultantes, procederemos a realizar la comparativa de los filtros con una representación. De ella podremos obtener información visual de que filtro está realizando un mejor filtrado.

Pasos cuarto bloque:

1. A continuación realizaremos la lectura de la última sección del script intentando comprender los comandos introducidos y buscando en la ayuda lo que no entendemos.

2. Ejecutaremos la última sección el script analizando los resultados en el apartado de variables de MATLAB, comprendiendo los valores obtenidos en función de las señales introducidas.

3. Por último al ejecutar la sección del script correspondiente, se nos abrirá un gráfico con dos representaciones, una sobre la otra. En la primera encontraremos la representación de todas las señales resultantes de los filtrados FIR con una leyenda diciéndonos, con cada respuesta, que filtro se ha utilizad,o indicándonos el color de la señal.
4. Y en la segunda representación, de forma análoga, se representan todos los resultados obtenidos del filtrado con los diversos filtros IIR empleados.

5. Se sacarán conclusiones de cual ha realizado mejor filtrado, y se realizarán cambios en la sección de generación de la señal a filtrar. Se tendrá en cuenta que la frecuencia de paso es de 2 KHz, viendo en cada caso cuál de los filtros es el mejor de las opciones a aplicar en función de la entrada introducida.

Anexos

Aprendizaje Autónomo:

Una vez terminada la realización y aprendida la metodología empleada, en el tercer experimento, volveremos al principio del bloque de "Comparación de filtros". Allí, podremos modificar los filtros a comparar, de tal manera que, según otro criterio de filtrado, como por ejemplo pasa banda, con otra frecuencias de paso y de stop a elecciones del alumno, y recorrer los distintos pasos de la herramienta, llegando de esta manera a otras conclusiones. También se podrá comparar otro tipo de filtros con otras finalidades y una señal de entrada distinta.

De esta manera, si el alumno realizase este experimento cuatro veces, una vez con cada tipo de filtro: pasa-bajos, pasa-altos, pasa-banda y para-banda, podría desarrollar una idea global, extrayendo las conclusiones de las gráficas comparativas, de que filtro usar en cada caso en función de las necesidades de filtrado y el tipo de señal de entrada introducida.

Se soluciona así, la toma de decisiones futuras de cómo saber qué tipo de filtro actuará mejor ante un caso concreto, pudiendo saber qué tipo de comportamiento tendrá en cada caso.

Nótese que si realizamos por ejemplo un filtrado donde la señal pasa sin alterarse, no seremos capaces de sacar conclusiones de cómo actúa cada filtro. Por lo que cuando los pongamos a prueba tendremos que utilizar señales de entrada que se acerquen a la frecuencia de paso y de stop, buscando las zonas más conflictivas, viendo cómo se modifican los resultados en cada caso. Se comprobará así cual mantiene mejor su comportamiento.

Anexos

Filtrado Digital de Señal Con DSP

Ejercicios Comparación de Filtros

Iñigo Rodríguez Martínez

Tutor: Antonio Moisés Zorzano Martínez

31/01/2017

Procesador de señal a emplear: **eZdsp TMS320VC5505 USB Stick**

UNIVERSIDAD DE LA RIOJA

A lo largo de este documento se plantean ejercicios a desarrollar por el alumno para continuar con el aprendizaje del filtrado mediante procesado digital, no limitándonos solo a los experimentos realizados y actividades guidas. Así se asentarán mejor los conocimientos y se adquirirán nuevos poniéndolo en práctica.

Anexos

Introducción:

Se plantean en este documento una serie de ejercicios más o menos abiertos en función de los casos, buscando un mayor manejo de la herramienta y la continuación del aprendizaje tras la parte guiada de los experimentos.

Se recomienda la lectura total de cada ejercicio antes de empezar a realizarlo ya que muchos de los objetivos pueden realizarse de manera simultánea.

Ejercicios planteados:

1. Partiendo de la batería de filtros presentes en el experimento. Se generarán diez filtros del mismo tipo que los utilizados anteriormente, en este caso pasa-alto.
   Los filtros tendrán una frecuencia de paso de 3500 Hz, ajustando los demás parámetros entre los otros filtros, para que todos tenga una atenuación y una frecuencia de stop similar.
   Hemos de tener en cuenta que el número de coeficientes de los filtros IIR no aumente de tal manera que, si hay que disminuir un poco las prestaciones, se realizará y se considerará en las observaciones, hasta donde podrán llegar las exigencias pedidas a estos filtros.

2. Modificaremos el script si es necesario, para representar el funcionamiento de estos filtros ante 3 señales de entrada distintas, incluyendo capturas de pantalla y un breve análisis de cuales presentan una respuesta mejor en qué casos.

3. Volveremos a generar los diez filtros anteriores en este caso con el propósito de realizar un filtrado pasa-banda, con una banda de paso de 1500 a 2500 Hz, ajustando los demás parámetros entre los otros filtros para que todos tenga una atenuación y una frecuencia de stop similar.
   Hemos de tener en cuenta que el número de coeficientes de los filtros IIR no aumente de tal manera que si en estos hay que disminuir un poco las prestaciones, se realizará y se considerará en las observaciones, hasta donde podrán llegar las exigencias pedidas a estos filtros.

4. Modificaremos el script si es necesario, para representar el funcionamiento de estos filtros ante 3 señales de entrada distintas, incluyendo capturas de pantalla y un breve análisis de cuales presentan una respuesta mejor en qué casos.

Anexos

5. Generaremos, por última vez, otra batería de diez filtros, en este caso para realizar un filtrado para-banda, con una banda de paro de 2200 a 2900 Hz, ajustando los demás parámetros entre los otros filtros para que todos tenga una atenuación y una frecuencia de stop similar.
   Hemos de tener en cuenta que el número de coeficientes de los filtros IIR no aumente de tal manera que, si hay que disminuir un poco las prestaciones, se realizará y se considerará en las observaciones, hasta donde podrán llegar las exigencias pedidas a estos filtros.

6. Modificaremos el script, si es necesario, para representar el funcionamiento de estos filtros ante 3 señales de entrada distintas, incluyendo capturas de pantalla y un breve análisis de cuales presentan una respuesta mejor en qué casos.

7. Se realizará un análisis detallado, revisando los ejercicios realizados anteriormente (2, 4, 6), donde se sacarán las conclusiones de los experimentos realizados, con una visión global de que tipos de filtros funcionan mejor en qué casos, señalando las señales de entrada introducidas para la adquisición de esos resultados.

8. Generación de una tabla comparativa de los resultados analizados en el ejercicio 7.

Anexos

Filtrado Digital de Señal Con DSP

Código reseñable del experimento

Iñigo Rodríguez Martínez

Tutor: Antonio Moisés Zorzano Martínez

31/01/2017

Procesador de señal a emplear: **eZdsp TMS320VC5505 USB Stick**

UNIVERSIDAD DE LA RIOJA

En este documento se introducirá el código reseñable de los programas que merezca la pena destacar de tal manera que queden reflejados en el documento.

Anexos

Código Matlab:

- Generación de señal a procesar

```
%% Generación de la señal a procesar
Fs = 24000;      % Frecuencia de muestreo
Ts = 1/Fs;       % Numero de muestras
F = 2000;        % Frecuencia de la senoidal
Amp = 5;         % Amplitud de la senoidal
sinewave = Amp*sin(2*pi*F*(0:Ts:1));          % Generación de la senoidal
noise = sqrt(0.1).*randn(1,length(0:Ts:1));   % var = 0.1 Ruido blanco
xn = sinewave%+noise;                  % Suma del ruido a la señal
xn_int = xn(1:256);                    % Nos quedamos con las 256 primeras muestras
plot(xn_int); axis([1 256 -Amp Amp]); % Visualizamos
ylabel('Amplitude'); xlabel('Sample index, n'); % Etiquetas gráfico
fid = fopen('xn_int.dat','w');    % Permiso de escritura en xn_int.dat
fprintf(fid,'%4.0f\n',xn_int);    % Guarda en formato entero xn_int
fclose(fid);                      % Cierra el fichero xn_int.dat
z=linspace(1,256,256);            % Crea un vector con un numero de valores igual
                                  % al numero de muestras a representar

```

Ilustración 63: Generación señal completa matlab

- Filtrado a partir de los coeficientes exportados

```
%% Filtrado de la señal (FIR)
% y=filter(b,a,x) Siendo:
% b numerador de los coeficientes del filtro
% a denominador de los coeficientes del filtro
% x señal a filtrar

% introduciremos en el siguiente apartado, si solo tiene numerador,
% colocaremos la variable sustituyendo la "b", y dejaremos "a" en 1, y si
% no rellenaremos con las dos variables guardadas numerador y denominador
y1=filter(Num1,1,xn_int);
y2=filter(Num2,1,xn_int);
y3=filter(Num3,1,xn_int);
y4=filter(Num4,1,xn_int);
y5=filter(Num5,1,xn_int);
%% Los Filtros IIR: Obtenidos en la fdatool los tendremos en forma de:
% Una matriz SOS y una ganancia G, la cual deberemos procesar para dejarlo
% de forma similar a los anteriores y poderlos comparar.
[b6, a6] = sos2tf(SOS1, G1);
y6=filter(b6,a6,xn_int);
[b7, a7] = sos2tf(SOS2, G2);
y7=filter(b7,a7,xn_int);
[b8, a8] = sos2tf(SOS3, G3);
y8=filter(b8,a8,xn_int);
[b9, a9] = sos2tf(SOS4, G4);
y9=filter(b9,a9,xn_int);
[b10, a10] = sos2tf(SOS5, G5);
y10=filter(b10,a10,xn_int);
```

Ilustración 64: Filtrado de las señales

Anexos

- Representación resultados:

```
64      %% Representación
65 -    subplot(2,1,1)
66 -    plot(z, y1, 'r-',z, y2, 'b-',z, y3, 'g-',z, y4, 'y-', z, y5, 'black-')
67 -    title('Comparación FIR'); xlabel('Muestras'); ylabel('Amplitud');
68 -    legend('FIR Equiripple','FIR GEquiripple','FIR LeastPN','FIR LeastSquares','FIR WindowK');
69 -    subplot(2,1,2)
70 -    plot(z, y6, 'r-',z, y7, 'b-',z, y8, 'g-',z, y9, 'y-', z, y10, 'black-')
71 -    title('Comparación IIR'); xlabel('Muestras'); ylabel('Amplitud');
72 -    legend('IIR Butterworth','IIR Chebyshev','IIR Eliptic','IIR LeastPN','IIR MaximallyFlat');
```

Ilustración 65: Representación resultados matlab

Anexos

Anexo 5:

Filtrado Digital de Señal Con DSP

Implementación de la Interfaz

Iñigo Rodríguez Martínez

Tutor: Antonio Moisés Zorzano Martínez

31/01/2017

Procesador de señal a emplear: **eZdsp TMS320VC5505 USB Stick**

UNIVERSIDAD DE LA RIOJA

A lo largo de este documento se expone el código más relevante introducido en Netbeans para que quede visible dentro del documento.

Anexos

Definición:

```
6   package tfg;
7
8   /**
9    TFG:    Procesado Digital de Señal
10            Autor:    Iñigo Rodríguez Martínez
11            Tutor:    Antonio Zorzano Martínez
12                      Logroño (La Rioja) España Noviembre del 2016 */
    import java.net.*;
    import java.awt.geom.*;
    import java.awt.*;
    import java.awt.event.*;
17  import java.io.IOException;
    import javax.swing.*;
    import javax.swing.event.*;
    import javax.swing.JTextField;
    import java.io.File;
    import static java.lang.Math.log;
23
24  public class Ventana extends javax.swing.JFrame {
25
26      /**
27       * Creates new form Ventana
28       */
29      public Ventana() {
30          initComponents();
31      }
32
33      /**
34       * This method is called from within the constructor to initialize the form.
35       * WARNING: Do NOT modify this code. The content of this method is always
36       * regenerated by the Form Editor.
37       */
```

Ilustración 66: librerías java y creación del marco

Botón de salida:

```
38      @SuppressWarnings("unchecked")
39      Generated Code
1192
1193    private void botonEXITActionPerformed(java.awt.event.ActionEvent evt) {
1194        System.exit(0);
1195    }
```

Ilustración 67: Botón de salida java

Anexos

Handler de los botones:

```
1197    private void botonCFmatlabActionPerformed(java.awt.event.ActionEvent evt) {
1198        try {Runtime.getRuntime().exec("rundll32 url.dll,FileProtocolHandler "
1199                +"C:\\Archivos_TFG\\Archivos_Matlab\\CF\\FDA_tool.m");
            } catch (IOException e) {e.printStackTrace();
1201    }
1202    }
1203
1204    private void botonCFguiaActionPerformed(java.awt.event.ActionEvent evt) {
1205        try {Runtime.getRuntime().exec("rundll32 url.dll,FileProtocolHandler "
1206                +"C:\\Archivos_TFG\\Documentos\\CF\\CF_Guia.pdf");
            } catch (IOException e) {e.printStackTrace();
1208    }
1209    }
1210
1211    private void button20ActionPerformed(java.awt.event.ActionEvent evt) {
1212        try {Runtime.getRuntime().exec("rundll32 url.dll,FileProtocolHandler "
1213                +"C:\\Archivos_TFG\\Archivos_Matlab\\IIR\\Coeficientes_Generados");
            } catch (IOException e) {e.printStackTrace();
1215    }
1216    }
1217
1218    private void botonInformacioGeneralActionPerformed(java.awt.event.ActionEvent evt) {
1219       try {Runtime.getRuntime().exec("rundll32 url.dll,FileProtocolHandler "
1220               +"C:\\Archivos_TFG\\Documentos\\Introduccion\\Informacion_Bloque_1.pdf");
            } catch (IOException e) {e.printStackTrace();
```

Ilustración 68: Manejador de archivos

Ventana de configuración de eventos y propiedades:

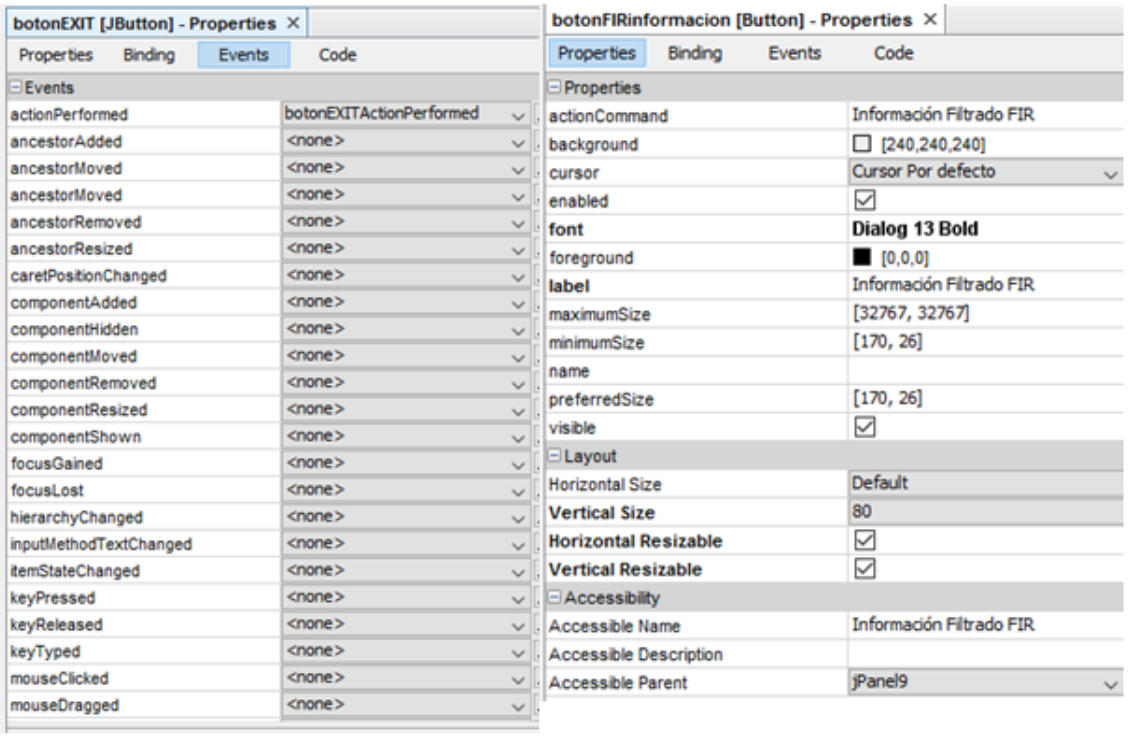


Ilustración 69: Configuración elementos java

6 – Pliego de condiciones:

El autor de este proyecto ha cursado los estudios de Grado de Ingeniería Electrónica Industrial y Automática en la Universidad de La Rioja, cumpliendo con la normativa establecida por la Escuela Técnica Superior de Ingeniería Industrial en la normativa de trabajo fin de grado.

El objeto de este Pliego de Condiciones es recoger y establecer todas las disposiciones técnicas, administrativas y económicas, y las normativas que ha de regir este proyecto.

El diseño de este proyecto y todas sus características han sido descritos en detalle en la memoria del proyecto y sus anexos.

Las condiciones que se especifican en este documento tratan de cumplir con la calidad esperada para este proyecto. En caso de no realizarse según estas condiciones, el proyectista no se responsabilizará de los fallos o averías que puedan ocasionarse en su funcionamiento, y los problemas derivados repercutirían sobre terceras personas.

6.1 - Condiciones Generales:

Este TFG sigue los reglamentos y normativas electrónicas vigentes. Una vez terminado el proyecto se podrán llevar a cabo modificaciones pero siempre bajo la propia responsabilidad del que la realiza.

La propiedad intelectual del autor y director del Trabajo Fin de Grado se regirá por el Real Decreto Legislativo 1/1996, de 12 de abril, por el que se aprueba el texto refundido de la Ley de Propiedad Intelectual.

El proyecto contiene los siguientes documentos:

- Un Índice General, que indica la página donde comienza cada una de las secciones dentro de este documento.
- Una Memoria, donde se consideraran todos los aspectos técnicos necesarios para el entendimiento del proceso de desarrollo de este TFG.
- Una serie de Anexos, donde se recogen la documentación contenida en la herramienta que es de especial interés para ampliar la descripción detallada de los componentes del sistema.
  Además, contiene las secciones de código no visibles desde la herramienta.

Pliego de condiciones

- Un Pliego de Condiciones. Dicho documento, donde se encuentra esta clasificación es en el que se establecen las diferentes condiciones técnicas, económicas y administrativas para que proyecto pueda materializarse, evitando posibles malinterpretaciones.
- Presupuesto donde se recoge el coste de todos los componentes utilizados y la suma total de costes, que, junto a las horas invertidas por el proyectista, dará el coste final del proyecto.

Pliego de condiciones

6.2 – Normativa y reglamentación:

El proyecto estará regido por la normativa internacional:

El único dispositivo hardware que emplearemos en este proyecto, ha sido elegido de una empresa de reconocimiento internacional y uno de los principales fabricantes de microprocesadores, como es Texas Instruments.

La normativa del dispositivo se encuentra en la página web del fabricante, y los planos del mismo en el Datasheet incluido en la bibliografía.

No se mencionan las nomas cumplidas por este dispositivo en este proyecto, ya que no es propiedad del mismo, ni del autor, ni de la Universidad.

Pliego de condiciones

6.3 – Condiciones facultativas:

6.3.1 – Dirección:

La dirección del desarrollo de la herramienta ha sido llevada a cabo por el ingeniero proyectista y el director de proyecto asignado.

Una vez terminada la herramienta, ésta podrá ser utilizada por cualquier persona con conocimientos básicos sobre el sistema y sus componentes.

En caso de avería o pérdida de información por una utilización incorrecta de la herramienta, el ingeniero proyectista o la persona en la que haya delegado la dirección del proyecto, quedan exentas de responsabilidad.

6.3.2 – Libro de órdenes:

El uso de la herramienta del proyecto se realizará siguiendo el siguiente orden de prioridad en el caso de encontrar contradicciones.

- Memoria
- Guías

6.3.3 – Modificaciones:

Si fuera necesario realizar alguna modificación en el presente proyecto para generar una futura publicación o generar una actualización publica, deberá comunicarse con anterioridad a su realización al autor, o en su defecto al Director de proyecto, quién deberá dar la correspondiente autorización.

En caso de realizarse modificaciones en la herramienta que no hayan sido previamente comunicadas y autorizadas por el autor y el Director de proyecto, las consecuencias que dichos cambios puedan ocasionar serán de total responsabilidad del usuario que las realice.

Respecto a los cambios en la herramienta realizados por el propietario de la misma, no serán tratados de forma especial y, en ningún caso, quedan eximidos de la autorización del autor y del Director de proyecto.

Pliego de condiciones

6.4 – Condiciones de materiales y equipos:

A continuación se detallan las condiciones tanto hardware como software, con las que hay que contar, para hacer uso de la aplicación del proyecto.

6.4.1 – Condiciones técnicas de los equipos

Todos los materiales y componentes, utilizados en este TFG, deben cumplir con las especificaciones técnicas que aparecen descritas durante la memoria y en el pliego de condiciones.

En cuanto a la tarjeta de sonido del PC donde se ejecuta el programa, y que esta fuera de control del autor, deberá tener un correcto funcionamiento para la realización de las diversas tareas que la requieran.

Así mismo, la tarjeta DSP, deberá estar en perfecto estado para garantizar un funcionamiento correcto de los programas a ejecutar y de la aplicación en sí misma en todas las funcionalidades de la herramienta que así lo requieran.

Para el desarrollo y uso de la herramienta se deberá disponer de un PC, con una tarjeta de sonido formando parte del mismo, que será el encargado de ejecutar los programas utilizados.

El software necesario en el que se ha probado esta herramienta corresponde a:

- Windows 7 (64 bits).
- Matlab R2015a.
- Code Composer Studio V 4.0.
- SoundCard Studio V 1.46.

Con las correspondientes licencias de uso los que así lo requieran.

La herramienta, generalmente, debería mantener sus funcionalidades con las versiones posteriores de las empleadas para su creación.

Pliego de condiciones

6.5 – Condiciones Económicas:

6.5.1 - Errores en el proyecto:

Si existiese algún tipo de error en la herramienta, se comunicará inmediatamente al autor del mismo y se le informará con detalle de los errores encontrados.

Además si al encontrar un error, éste influyese directamente sobre los resultados, no se podrán verificar los mismos, de tal manera que no se deberían sacar conclusiones sobre los resultados obtenidos, ya que estos pueden ser equívocos y crear ideas erróneas en el aprendizaje.

6.5.2 – Comercialización:

Este proyecto dispondrá de uso gratuito para uso educativo, por lo cual los estudiantes de esta universidad tendrán posibilitado su uso de manera libre.

En cuanto a fines educativos, dispondrá de uso libre a nivel de usuario, para que cualquier persona que desee aprender sobre este tema tenga la posibilidad de realizarlo sin ningún impedimento.

7 - Bibliografía y fuentes:

- Website TMS320C5505: http://www.ti.com/tool/TMDX5505EZDSP#descriptionArea
- Datasheet C5505: http://www.ti.com/lit/ds/sprs503b/sprs503b.pdf
- Website TLV320AIC3204: http://www.ti.com/product/TLV320AIC3204/description
- Datasheet TLV320AIC3204: http://www.ti.com/lit/ds/symlink/tlv320aic3204.pdf
- Matlab información 1: https://es.mathworks.com/products/matlab.html
- Matlab información 2: https://es.wikipedia.org/wiki/MATLAB
- Gráficas Matlab: http://www.mat.ucm.es/~rrdelrio/documentos/rrrescorial2002.pdf
- Website Netbeans: https://netbeans.org/community/releases/82/
- Netbeans Información: https://es.wikipedia.org/wiki/NetBeans
- Website SoundCard Scope: https://www.zeitnitz.eu/scms/scope_en?mid=4.01
- Website Code Composer Studio: http://www.ti.com/tool/CCSTUDIO
- Wiki CCS v 4.0: http://processors.wiki.ti.com/index.php/Category:Code_Composer_Studio_v4
- CCS información 1: http://www.ti.com/ww/mx/multimedia/webcasts/Code_Composer_Studio-4x_01-10-2010.pdf
- CCS información 2: https://en.wikipedia.org/wiki/Code_Composer_Studio
- Introducción al filtrado: http://www.dtic.upf.edu/~egomez/teaching/sintesi/SPS1/Tema7-FiltrosDigitales.pdf
- Filtros FIR información1: http://www.ingelec.uns.edu.ar/pds2803/materiales/cap10/12-cap12.pdf
- Filtros FIR información2: http://www.fimee.ugto.mx/profesores/arturogp/documentos/Filtrado%20Digital/Lectura%203_Filtrado_Digital.pdf
- Filtros FIR información3: http://www4.tecnun.es/asignaturas/tratamiento%20digital/tema8.pdf
- Técnicas de diseño FIR: http://www4.tecnun.es/asignaturas/tratamiento%20digital/TEMA9/tsld007.htm

Bibliografía y fuentes

- Filtrado FIR, IIR: http://www.ehu.eus/procesadoinsvirtual/T6_filtros%20iir%20y%20fir1.html
- Transformación bilineal: https://es.wikipedia.org/wiki/Transformaci%C3%B3n_bilineal
- Filtros IIR información1: https://docs.google.com/viewer?a=v&pid=sites&srcid=ZGVmYXVsdGRvbWFpbnxlbGVjdGl2YTJ1bmlwYXp8Z3g6NThjNmRkYjZhOWJkOGI0MA
- Filtros IIR información2: http://www4.tecnun.es/asignaturas/tratamiento%20digital/tema8.pdf
- Filtros IIR información3: http://ocw.uv.es/ingenieria-y-arquitectura/filtros-digitales/tema_4_diseno_de_filtros_iir.pdf
- Filtros IIR información4: https://es.wikipedia.org/wiki/IIR#Dise.C3.B1o_de_filtros_IIR
- Filtros butterworth: http://filtrosbutterworthw.blogspot.com.es/
- Apuntes de procesado digital de señal de la Universidad de la Rioja.
- Libro: Real-Time Digital Signal Processing. SEN M.KUO | BOB H.LEE: catálogo de la Universidad de la rioja. (Como utilizar el DSP para filtrado) ISBN:978-1-118-41432-3 http://catalogo.unirioja.es/cgi-bin/abnetopac?TITN=368493
- Manipulación de audio Matlab: https://es.slideshare.net/ricknag/mat-lab-manipulacin-de-seales-de-audio
- Ejemplos de filtrado: http://agamenon.tsc.uah.es/Asignaturas/it/tds/apuntes/practica_sfd_2008_09.pdf
- Documentación filtrado unipaz: https://sites.google.com/site/electiva2unipaz/procesamiento-digital-de-senales/diseno-de-filtros-digitales

8 - Presupuesto:

Para llevar a cabo este proyecto se ha elaborado el siguiente presupuesto, donde quedan reflejadas las horas presupuestadas con su precio correspondiente.

Concepto	Horas (h)	Precio hora (€/h)	Total(€)
Kit DSP de Texas Instruments			50 €
Investigación anterior a la realización	40 h	18 €/h	720 €
Desarrollo de la herramienta	80 h	35 €/h	2.800 €
Test	14 h	25 €/h	350 €
Redacción de documentos propios de la herramienta	22 h	18 €/h	396 €
Redacción de documentos propios de la memoria	26 h	18 €/h	468 €
Total			**4.784 €**

Ilustración 70: Presupuesto

*Todos los precios incluyen IVA.

Firmado: Iñigo Rodríguez Martínez

LOGROÑO, 27 de febrero de 2017.